ALM - Application Lifecycle Management for Developers

Duration: 4 Days **Course Code: GK3346**

Overview:

Visual Studio 2012 and Visual Studio Team Foundation Server 2012 help software development teams successfully deliver complex software solutions. Learn how Visual Studio and Team Foundation Server enable you to enforce best practices for software development and to develop better quality software. This course uses the very latest (2012) versions of Visual Studio and Team Foundation Server.

You'll get answers to these questions: How do software development teams successfully deliver complex software solutions?How do we improve visibility and monitor the progress and health of large projects?How do we optimally configure the source control system to suit our requirements?How do we decide on the best branch plan to use to manage ou development projects?How do we set up an automated build pipeline to support Continuous Integration (CI)?How do we use Test-Driven Development (TDD) to deliver demonstrably higher quality?How do we reduce the overhead of writing unite tests and ensure that they are effective?How do we improve the extensibility, maintainability and testability of our code?How do we test code that uses external systems suach as databases and remote services?How do we easily pinpoint and solve issues that occur once the application is live?How do we avoid problems with performance, reliability, security and maintainability?How do we improve the robustness of code that we write for other teams to consume?

Objectives:

- Learn how to configure Team Foundation Server to support your software development process
- Use Work Item Tracking to support software development using Scrum, Agile, CMMI and Kanban
- See how to take maximum advantage of the source control system and benefit from workspaces
- Demystify branch visualisation, change tracking and how to create custom check in policies
- Compare the different types of automated build and see how to customise the build process
- Investigate how to craft good unit tests and how to manage very large suites of unit tests
- Build data-driven unit tests to improve testing and use code coverage to detect weak areas
- Appreciate the benefits of using Dependency Injection (DI) and Inversion of Control (IoC)
- Explore how to generate and use fakes, stubs, spies and mocks by using mocking frameworks
- Transform problem solving by quickly locating problems in development, test and production
- Perform automated best practice reviews and implement custom rules for company standards
- Understand how to easily verify correct method and object use at compile time and run timea

Content:

Day 1

Introduction to Visual Studio and Team Foundation Server

Work Item Tracking in Scrum, Agile, CMMI and Kanban

Day 2

Introduction to Source Control

In this module, we will look at the source control system. Visual Studio and Team Foundation Server support the concept of a repository, a server-side store for managing source code and other project artefacts. Team members check items into and out of source control using workspaces, which store and manage client-side copies of server-side files. This module addresses all these concepts in detail, in addition to looking at conflict resolution and TF.exe, the Team Foundation Version Control Tool.

Branching, Merging and Shelving

The source control repository supports the concept of branching, which is an isolation mechanism that allows multiple version of a codebase to be managed, worked on and versioned independently. This allows developers, for example, to start work on new features while the main codebase is being stabilised ready for release. This module explores branching and reviews a number of strategies that can be used when devising your branch plan. It will also cover merging, which is the process of taking changes from one branch and merging them into another and shelving, a way to save changes on the server that are not yet ready to be incorporated into the live codebase. Branch visualisation, change tracking and merge conflict resolution are also covered. Finally, check in policies are examined as a means of controlling what is checked in, alo

Team Foundation Build

- On large software teams, it is often desirable to set up a build environment where builds of the project are performed automatically based on some trigger, for example when doing continuous integration or nightly builds. This module investigates how Team Foundation Build supports fully automated builds, from the build controllers and

Day 3

Test-Driven Development and Unit Testing

Unit testing is now considered by most developers to be an essential part of writing high quality software and it is now compulsory in many organisations. Test-driven development (TDD) is the next logical progression from unit testing, which delivers demonstrably higher levels of code quality than simple unit testing. This module takes a detailed look at unit testing and TDD, from the real-world benefits to how to craft good unit tests (and how to avoid writing bad ones). It also explains the AAA pattern, the red-green-refactor approach to TDD, the continuous test runner and how to manage large suites of unit tests so that the right tests can be run frequently and quickly.

Data-Driven Unit Testing, Code Coverage

Developers who practice unit testing and TDD often find themselves needing to test a behaviour in a number of different ways.

Doubles and Mocking

- When unit testing code, it is frequently necessary to test code that uses external resources, such as the file system, databases and remote services. To make this possible, a substitute for the Depended On Component (DOC) is injected into the System Under Test (SUT) that provides the necessary interface but which doesn't perform any e

Day 4

IntelliTrace

Code Analysis

The cost of correcting a defect in an application rises very significantly throughout the development cycle, hence the well-known expression "Early bugs are cheap bugs". One of the most effective ways to identify and eliminate defects early on is to conduct frequent code reviews. Unfortunately, conventional code reviews are both time consuming and resource intensive, meaning that they are rarely carried out as often as they should be. Visual Studio includes a code analysis engine that is able to perform a code review on demand or on every build of an application. It is able to identify many common problems that can occur, including issues with performance, reliability, security, maintainability, best practice violations and much, much more. This module demonstrates how to configure code analysis to suit specific needs, how to under

Code Contracts

The advantages of implementing method preconditions, postconditions and object invariants are well understood. However, comprehensively implementing these properly can be difficult and time consuming. This module looks at how code contracts, which are part of the .NET Framework, can implement preconditions, postconditions and object invariants simply, quickly and effectively. Code contracts also provide a major advantage over manually implemented equivalents, namely that code contracts can be verified at compile time as well as at run time. This provides immediate and comprehensive feedback on all contract violations without needing to test the

While this can be achieved by writing a number of very similar unit tests, a better way is to create a data-driven unit test, which allows a single test to execute many times with different test values. This module examines how to create and configure data-driven unit tests. It also looks at code coverage, which is a way to determine easily how effective the unit test suite is and to identify any areas in the software which could benefit from the creation of additional tests.

Dependency Injection and Inversion of Control

Many developers have heard about the clear advantages of adhering to the SOLID principles of good design, but not all are fluent in applying them. Dependency injection is not only key to writing SOLID code, but it can also help implement late binding, improve extensibility, assist large teams practicing parallel development and improve maintainability. Importantly, it can also significantly improve the testability of code. This module not only looks at the different forms of dependency injection and at how to use dependency injection correctly, but also explains how to use an Inversion of Control (IoC) container to handle dependency management and complex construction automatically. Configuring the IoC container in code and in the application configuration file, which allows the application to be reconfigured without the need for r

application. The configuration of the tooling and of the static and runtime analysis are also covered.

Further Information:

For More information, or to book your course, please call us on 00 20 (0) 2 2269 1982 or 16142

training@globalknowledge.com.eg

www.globalknowledge.com/en-eg/

Global Knowledge, 16 Moustafa Refaat St. Block 1137, Sheraton Buildings, Heliopolis, Cairo