

Programmera C#: utökad funktionalitet - 4 dagar

Effektivisera Entity Framework-applikationer

kurser 973

- Du får lära dig att**
- Effektivisera datacentrerade applikationer med utökad funktionalitet i C# och Entity Framework (EF)
 - Lambda-uttryck och extension methods (utökningsmetoder) för hantering av applikationslager
 - Utnyttja Language Integrated Query, LINQ-nyckelord, för att filtrera och ordna data
 - Överföra komplex logik med de generiska delegaterna `Func<T,R>`
 - Generera dynamiska numreringar med `IEnumerable<T>` inom utvecklarens skrivna generiska klasser
 - Använda LINQ för att förenkla XML-processer
- Sammanfattning** C# har utvecklats till ett komplett datahanteringsspråk. Utökad funktionalitet i .NET hjälper programmerare att effektivisera kod, öka produktiviteten och förbättra applikationers prestanda. På den här kursen får du använda den utökade funktionaliteten i C#, inklusive LINQ och EF, för att effektivt integrera objektorienterings- och datahanteringsfunktioner.
- Vem bör delta** Erfarna C#-programmerare som vill förbättra sina kunskaper i mjukvaruutveckling genom att använda utökade språkfunktioner, i synnerhet Language Integrated Query (LINQ), tillsammans med Entity Framework. Erfarenhet av C#-programmering i nivå med kurs 419, "C#-programmering", förutsätts.
- Praktiska övningar** En pågående fallstudie ger erfarenhet av att tillämpa utökad funktionalitet i C#, bland övningarna ingår:
- Använda auto-egenskaper och objektinitialiserare
 - Skriva lambda-uttryck och utökningsmetoder
 - Implementera datalagret med LINQ och EF
 - Använda DataContext-objekt och LINQ-nyckelord för att komma åt och uppdatera en databas
 - Mappa entitetsklasser med O/R Designer
 - Skapa dynamiska enumerationer med `yield`
 - "Parsa" XML-document med LINQ
 - Åtkomst till lagrade processer med C#/LINQ

Programmera C#: utökad funktionalitet - 4 dagar

Effektivisera Entity Framework-applikationer

kurser 973

Introduktion

- Genomgång av objektorienterad programmering
- Implementera ett gränssnitt
- Befintliga tekniker för dataåtkomst
- Generiska och icke-generiska samlingar

Använda språkfunktioner i C#

Genvägar i språket

- Tillämpa auto-implementerade egenskaper
- Dra nytta av underförstådda typdeklarationer

Effektivisera programlogik

- Exemplifiera enhetsobjekt
- Förenkla konstruktion med objekt-initialisera
- Anonym objekt-konstruktion

Använda utökningsmetoder för datahantering av mellanlager

Filtrera och ordna data med lambda-uttryck

- Konstruera lambda-uttryck
- Jämföra delegater och lambda-uttryck
- Anropa **Count**, **Reverse**, **Union**, **Except** och andra utökningsmetoder
- Skicka typer och funktionalitet till metoder

Tillämpa utökad funktionalitet

- Göra kod mer flexibel med delegater
- Parameterisera delegater och lambda-uttryck
- Effektivisera kod med de generiska delegaterna **Func<T,R>**

Använda LINQ-frågenyckelord

Syntax och semantik

- Koda LINQ-frågor
- Ordna data och objekt
- Filtrera med **from**, **where**, **orderby** och **group**

Loopa genom kollektioner

- Överföring mellan C# och LINQ med hjälp av **IEnumerable<T>**
- Konvertera från **IEnumerable<T>** till **List<T>**

Anpassa LINQ och EF

Jämföra ADO.NET med LINQ

- **DataSet**-objekt jämfört med generiska listor
- **SqlDataReader** jämfört med **IEnumerable<T>**

- Behandla datalager-information

Ansluta till och läsa från en databas

- Fastställa en **ObjectContext**
- Ansluta till databaser via Entity Framework (EF)
- Korrelera entitetsklasser och datatabeller
- Bevara det objektorienterade paradigmet

Verktuget Object Relational Designer

- Mappa datatabeller till entitetsklasser
- Fastställa arvsförhållanden

Uppdatera en databas

- Infoga, uppdatera och radera data
- Felhantering och undantag
- Avsluta och rulla tillbaka transaktioner

Enumerationer och generiska klasser

Mer om enum

- Jämföra **IEnumerable<T>** och **IEnumerator<T>**
- Generera dynamiska listor
- Nyckelordet **yield**

Skriva generiska klasser

- Minska duplicerade klasser
- Implementera en generisk snabb enumerator

Bearbeta data med LINQ-frågor

Bearbeta data

- Kombinera och slå ihop liknande data med **group**
- Utföra inre, yttre och grupp-joins
- Generera data subsets med satsen **into**

Avancerade LINQ-tekniker

- Skapa anonyma frågeresultat
- Hämta sammansatta vyer med hjälp av nästlade **from**-satter
- Förbättra LINQ-frågor med hjälp av delegater och lambda-uttryck

Applicera LINQ på XML

Använda XML Namespace

- Ladda XML dynamiskt via webben
- Skapa och spara XElement-innehåll

Bearbeta XML

- Hämta dokument, element och attribut
- Tolka ett XML-dokument med hjälp av LINQ