SDFASFALK

ROTEIRO

CSHARP E .NET

Palavra Chave: CSharp e .NET
Tempo estimado: 10 minutos.
Objetivos do vídeo: Ensinar detalhadamente as passagens do .NET e C#
Título da aula: CSharp e .NET

SDC™

Sejam todos bem vindos a mais um episódio, aqui na sociedade do código.
Como vocês estão? Tudo bem?

Na aula de hoje vamos falar sobre o C# e o ecossistema .NET.

Ao final desta aula você será capaz de explicar as características do C#, assim como analisar todas as nuances do ecossistema .NET, então vamos lá.

O CSharp é uma linguagem de alto nível, tipada, compilada e com código gerenciado.

Agora vamos entender cada um desses termos:

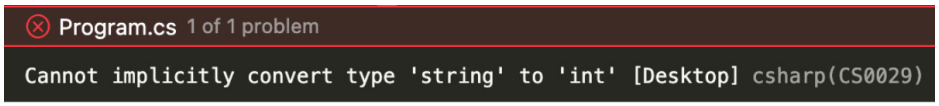
Tipada: Dizer que C# é fortemente tipada significa que seus objetos ou variáveis devem ser iniciadas informando qual o valor contido nelas, por exemplo:

Quando criamos uma variável inserindo a sintaxe ”var”, devemos obrigatoriamente informar o valor a ser atribuído à ela, como fizemos no exemplo anterior.

Caso façamos a inicialização da variável informando o tipo incorreto, receberemos um erro.

Aqui vamos criar uma variável do tipo int, e vamos atribuir o valor ”Binhara” a ela.

O que será exibido para nós é esse erro aqui:



O que está acontecendo é que a ferramenta está informando para nós que não é possível gerar esse tipo de conversão implicitamente, ou seja, o tipo int não aceita que façamos a inserção do tipo string para sua criação. E isso acontece devido a tipagem forte do Sharp.

O nosso próximo termo é compilada.

O que significa ser uma linguagem compilada?

Nas aulas 9 e 14 do programador profissional falamos um pouco sobre o que significa uma linguagem compilada e interpretada, então se você quiser acompanhar o conteúdo que tem lá também, será bom complemento

Seguindo adiante...

Quando falamos sobre uma linguagem compilada, queremos dizer que ela passará por um processo antes de ser transformada em um programa executável.

No caso do C# e de outras linguagens .NET, podemos dividir esse processo em 4 etapas, conforme demonstrado pela própria microsoft na documentação que deixarei nos detalhes deste vídeo.

Estas etapas são:

1. **Escolher um compilador**

Nessa parte do processo, o .NET conta com um recurso denominado Common Language Runtime (CLR), que fará toda a verificação do código que foi escrito, através do compilador selecionado. Nós falaremos mais sobre isso quando passarmos pela parte do significado do código gerenciado, que envolve o CLR.

O que acontece aqui é bem interessante, preste atenção.

Há uma especificação, que define as características do CLR, o ECMA-335.

O que essa especificação faz, é garantir que qualquer linguagem, que pretenda desenvolver aplicações para o .NET, contenha um compilador que siga essas predefinições.

Dessa forma, você consegue construir aplicações .NET utilizando C#, F#, Cobol, Ruby, Perl entre várias outras linguagens.

Isso é simplesmente sensacional!

Você precisa apenas de um compilador que obedeça às características definidas para o .NET, e ai pode programar a partir de várias linguagens,

Agora a etapa número dois.

2. Compilando código para Linguagem intermediária Microsoft.

Nesse momento vai ocorrer a conversão do código para uma linguagem intermediária, a (MSIL) de Microsoft Intermediate Language, ou apenas (IL) Intermediate Language.

A linguagem intermediária é um conjunto de funções para carregar, armazenar, iniciar métodos em objetos, contém também instruções lógicas e matemáticas, controle de fluxo, acesso a memória, manipulação de erros e outros recursos.

Para que o código seja executado a MSIL deve ser convertida em código específico de CPU, geralmente através de um compilador JIT (Just-In-Time).

Quando o compilador produz o MSIL, ele também gera metadados, que descrevem os tipos utilizados no código, incluindo as suas definições, as assinaturas dos seus membros, as referências a outros códigos e qualquer coisa que será utilizada em tempo de execução.

Feito tudo isso, durante a execução, o sistema localiza e extrai dos metadados as informações que necessitar.

vamos para o próximo passo.

3. Compilando o código da linguagem intermediária (MSIL) para a nativa.

Para que possamos executar o MSIL, ele deve ser compilado no CLR para código nativo, mas há uma particularidade:

Ele deve obedecer a arquitetura do computador de destino, e para isso o .NET possui duas maneiras de executar a conversão.

utilizando o JIT (Just-In-Time)
utilizando o Ngen.exe que é um gerador de imagem nativa.

Essas etapas estão bem definidas na documentação que disponibilizei pra vocês, então se quiserem dar uma olhada a fundo sobre como acontece, fiquem à vontade.

Agora, partiremos para a etapa final, que é a execução do código.

4. Executando o código.

Como já dito nos tópicos anteriores, durante esta execução o CLR e os metadados irão fornecer as informações que forem necessárias para a execução da aplicação.

E ela poderá acontecer das duas formas que foram apresentadas anteriormente.

Por fim, temos a aplicação compilada sendo executada.

Agora, vamos analisar a última parte do nosso trecho, que fala sobre **código gerenciado.**

Nós falamos sobre um código ser gerenciado, quando a execução dele depende de um gerenciador.

No caso do .NET, existe o runtime, e esse runtime é aquilo que falamos anteriormente o CLR.

além do desenvolvimento ficar mais rápido, nós evitamos uma série de problemas que poderiam ocorrer caso tivéssemos que organizar manualmente as questões da linguagem, como por exemplo, a alocação de memória.

Um problema bem comum que existia antigamente, era quando um programa tentava acessar um local da memória que estava reservado para o sistema do windows.

e esse problema gerava a famosa tela azul.

Com o gerenciador de código, nossa vida fica mais fácil já que não precisamos nos preocupar com questões de máquina, do trabalho de máquina, e podemos focar melhor no business do negócio.

Pessoal, essa foi nossa aula de hoje, espero que tenham gostado e caso tenham alguma dúvida deixem aqui nos comentários que vamos te ajudar, beleza?

Na próxima aula falaremos mais a fundo sobre o ecossistema .NET, algo que muita gente que trabalha com a tecnologia tem dificuldade em entender.

Nos vemos na próxima aula. Fui!

OBRIGADO.

SDC™