I Programa detalhado de Ambientes Virtuais de Execução

Horas	Título	Descrição	Bibliografia
2,5 Semanas	**Apresentação da Plataforma .NET**	Programação orientada ao Componente (POC). Requisitos das plataformas de execução modernas:, interoperabilidade, portabilidade, segurança, frameworks, serviços de *runtime*, controlo de versões.Modelos de execução *managed* e *unmanaged*. Common Language Infrastructure (`CLI`) e a implementação da Microsoft (CLR). Linguagem Intermédia (CIL). Sistema de tipos. Metadata. Comparação sumária com a plataforma Java. Construção de componentes: *assemblies* e módulos. Compatibilidade binária. Máquina virtual: carregamento e execução de *assemblies*; compilação JIT. Interoperabilidade entre linguagens. (exemplos em C++/CLI,VB.net, C#)	CLR via C#, Cap. 1 e 2 Essential .NET, Cap. 1
1 Semanas	**Conceitos fundamentais do modelo de tipos: Tipos, Objectos, Valores**	Apresentação do Sistema de Tipos do CLI (CTS- Common Type System). Interoperabilidade entre linguagens – CLS. Categorias de tipos: Tipos Valor e Tipos Referência. Conversão de tipos: instruções `castclass` e `isinst`. Conversão entre tipos numéricos com e sem teste de *overflow*. Conversão entre tipos valor e correspondente tipo referência: `box` e `unbox`. Método Equals em tipos valor. *Overload* de operadores.	CLR via C# Cap 4 e 5 Essential .NET, Cap. 3,4
1,5 Semanas	**Estrutura de Tipos**	Membros de tipos: construtores de tipo e de instância. Construtores de instância em C#. Métodos virtuais. Atributo *new* aplicado a métodos virtuais e não virtuais. Constituição das tabelas de métodos. Passagem de parâmetros por valor e referência. Atributos out e ref na linguagem C#. Propriedades. Interfaces, *Arrays* e Enumerados.	CLR via C# Cap. 6 a 9, 12, 13, 14 Essential .Net Cap 5,6
1 Semana	**Delegates e Eventos**	Delegates. Comparação com ponteiros para função da linguagem C. Callbacks. As classes Delegate e MultiCastDelegate. Interfaces versus delegates. Eventos. registo costumizado. Exemplos	CLR via C# Cap. 15, 10
1 Semana	**Genéricos**	Genéricos. Interfaces e Coleccções genéricas. Constrangimentos (*constraints*). Implementação de genéricos. Tipos abertos e fechados. Métodos genéricos. Tipos *nullable.*	CLR via C# Cap. 16, 18
1 Semana	**Gestão de Memória**	O Heap. Libertação automática de memória. Finalização de objectos; finalização determínistica; Destrutores em C#. Pattern Dispose.	CLR via C# Cap. 20
1 Semana	**Reflexão**	Introspecção. Modelo computacional de utilização. Exemplos de utilização: persistência, serialização; Extensibilidade do modelo de metadados; atributos; atributos costumizados (*custom attributes*). Exemplos.	CLR via C# Cap. 22
1 Semanas	**Controlo de versões e partilha de componentes**	Assemblies privados versus Assemblies partilhados. Versões; *strong names*. Global Assembly Cache (GAC). Políticas de utilização de versões. Ficheiros de configuração.	CLR via C# Cap. 3 Essential .NET, Cap. 2
1 Semana	**Interoperabilidade com código *unmanaged***	Modos de Execução. Frames de transição. Passagem de parâmetros. P/Invoke. *Marshalling*.	Essential .NET, Cap. 10
1 Semana	**AppDomains**	Application Domains (AppDomains) . Comunicação entre AppDomains. *Marshaling*. Cópia por valor e referência.	Essential .NET, Cap. 8 CLR via C# Cap. 21

Objectivos de aprendizagem

1. Apresentação da plataforma .NET
    a. Capacidade de identificar os principais elementos de uma infra-estrutura para execução controlada de código, nomeadamente da infra-estrutura .NET;
    b. Compreensão da máquina de stack para execução de código intermédio, em particular a da infra-estrutura .NET;
    c. Compreensão da estrutura de um componente .NET (Assembly);
    d. Capacidade de lidar com ferramentas de linha de comando para compilar e analisar componentes .NET;
    e. Compreensão do processo de carregamento de assemblies e a sua ligação com a compilação Just-In-Time.

2. Conceitos fundamentais do modelo de tipos: Tipos, Objectos e Valores
    a. Compreensão das diferenças entre as duas categorias de tipos que constituem o CTS;
    b. Compreensão dos mecanismos que dão suporte, em tempo de execução, às operações de *cast* entre tipos referência;
    c. Compreensão do processo de conversão entre instâncias de tipos valor e instâncias dos respectivos tipos referência;
    d. Compreensão da semântica de comparação para tipos valor e tipos referência.

3. Estrutura de Tipos
    a. Capacidade de identificar os diferentes tipos de membros que podem fazer parte de um tipo do CTS;
    b. Capacidade de definir tipos com construtores de instância e de tipo, métodos estáticos e de instância (polimórficos e não polimórficos) e propriedades.
    c. Compreensão dos mecanismos que dão suporte, em tempo de execução, ao despacho dinâmico de métodos, via referências de classes e de interfaces;
    d. Compreensão do modelo de memória subjacente à passagem de parâmetros por referência.

4. Delegates e eventos
    a. Capacidade de comparar o mecanismo delegates com soluções funcionalmente equivalentes fora da plataforma .NET (eg: listeners de eventos na linguagem Java).
    b. Saber criar e usar delegates num programa C#;
    c. Compreensão da estrutura interna das instâncias de delegate;
    d. Compreensão da categoria de membro evento;
    e. Capacidade de definir, em C#, membros do tipo evento, incluindo o registo costumizado de handlers de eventos;
    f. Compreensão dos artefactos produzidos pelo compilador na compilação de membros do tipo evento (código intermédio e metadata).

5. Genéricos
    a. Capacidade de usar alguns dos principais tipos e métodos genéricos da *Base Classe Library*, em particular delegates e colecções;
    b. Capacidade de distinguir entre tipos abertos e tipos fechados e de comparar com os genéricos da linguagem Java;
    c. Capacidade de definir, em C#, tipos com(?) métodos genéricos; (constraints)
    d. Capacidade de utilização do tipo *Nullable* e compreensão da forma como este tipo é tratado pela máquina virtual nas operações de box e unbox.
    e. Capacidade de definir, em C#, iteradores *lazy*;

6. Garbage Collection
    a. Conhecer os algoritmos usados para efectuar a recolha automática de memória e saber identificar suas vantagens e desvantagens
    b. Compreensão dos mecanismos usados no CLR para efectuar a gestão automática de memória identificando pontos de suporte na metadata e no compilador JIT
    c. Compreensão do ciclo de vida de um objecto com finalizadores.
    d. Saber o modelo computacional para interação com o gestor de memória.
    e. Capacidade para escrever código adaptado à utilização do Garbage Colector

7. Reflexão
    a. Conhecer o modelo de classes de acesso à metadata.
    b. Capacidade para construir programas que lidam com entidades não conhecidas em tempo de compilação.
    c. Compreensão do mecanismo de extensibilidade da metadata e construção de aplicações que dele tiram partido
    d. Capacidade para construir programas extensíveis.

8. Deployment
    a. Conhecer as componentes do nome de um strong named assembly.
    b. Capacidade para criar assemblies strong named.
    c. Compreensão do mecanismo de identificação e binding usado no carregamento de assemblies strong named.

9. Interoperabilidade com código unmanaged
    a. Capacidade para construir aplicações que tiram partido de serviços nativos através do mecanismo de P/Invoke
    b. Compreensão da problemática relacionada com a invocação de código nativo, nomeadamente os aspectos de *marshaling* de parâmetros, convenções de chamada e interacção com o Garbage Colector
    c. Compreensão do conceito de modo de execução. Frames managed e unmanaged. Frames de transição.