IFRN

PROGRAMAÇÃO ORIENTADA A OBJETOS EM C#

Arquivos e Serialização

Prof. Gilbert Azevedo

Objetivos

- Utilizar arquivos de texto na programação de aplicativos
- Serializar de objetos nos padrões CSV, XML e Json

Arquivos Texto

- Arquivos Texto
  - É um repositório de strings armazenado no sistema de arquivos do computador e normalmente com extensão txt
  - Utiliza um padrão ou codificação para os caracteres que pode variar de acordo com o idioma utilizado
- Manipulação de Arquivos
  - O C# utiliza as classes StreamReader e StreamWriter para controlar a leitura e escrita de dados em um arquivo texto
  - Ambas as classes estão definidas no espaço de nomes System.IO

Escrevendo um Arquivo Texto

- A classe StreamWriter controla um fluxo de saída
  - Construtor: Abre o fluxo de escrita
  - WriteLine: Escreve uma linha no arquivo
  - Close: Fecha o fluxo garantindo a escrita correta dos dados

```
StreamWriter f = new StreamWriter("teste.txt");
string s = Console.ReadLine();
while (s != "fim") {
  f.WriteLine(s);
  s = Console.ReadLine();
}
f.Close();
```

Lendo um Arquivo Texto

- A classe StreamReader controla um fluxo de entrada
  - Construtor: Abre o fluxo de leitura
  - ReadLine: Lê uma linha do arquivo
  - Close: Fecha o fluxo e libera o recurso para o sistema operacional

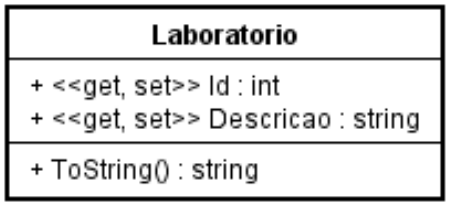
```
StreamReader f = new StreamReader("teste.txt");
string s = f.ReadLine();
while (s != null) {
  Console.WriteLine(s);
  s = f.ReadLine();
}
f.Close();
```

Serialização

- Serialização
  - Processo para representar um objeto em um formato usado no armazenamento ou transmissão de dados
  - Resultado pode ser salvo em arquivo de texto seguindo algum padrão
- Arquivos CSV (Comma Separeted Values)
  - É um arquivo texto com valores separados por vírgula usado em planilhas
- Arquivo XML (eXtensible Markup Language)
  - É um arquivo que usa Tags para representar as informações armazenadas
  - Vantagem: armazena o dado e o meta-dado
- Arquivo Json (JavaScript Object Notation)
  - Mais “leve” que o XML e bastante utilizado em aplicações na web

Exemplo

- Sistema de Monitoramento de Laboratórios
  - Cadastro de Laboratórios




```
public class Laboratorio {
  public int Id { get; set; }
  public string Descricao { get; set; }
  public override string ToString() {
    return $"{Id} - {Descricao}";
  }
}
```

Arquivo CSV

- Para escrever arquivos CSV
  - Utilizar a classe StreamWriter para escrever o arquivo
  - Programar a lógica para serializar o objeto em texto

```
List<Laboratorio> labs = new List<Laboratorio>();
labs.Add(new Laboratorio {Id = 1, Descricao = "Lab01"});
labs.Add(new Laboratorio {Id = 2, Descricao = "Lab02"});

StreamWriter f = new StreamWriter("Laboratorio.csv");
foreach(Laboratorio lab in labs)
  f.WriteLine($"{lab.Id};{lab.Descricao}");
f.Close();
```

Laboratorio.csv

```
1    1;Lab01
2    2;Lab02
```

	A	B
1	1	Lab01
2	2	Lab02

Arquivo CSV

- Para ler arquivos CSV
  - Utilizar a classe StreamReader para ler o arquivo
  - Programar a lógica para de-serializar o texto em objeto

```
List<Laboratorio> labs = new List<Laboratorio>();

StreamReader f = new StreamReader("Laboratorio.csv");
string s = f.ReadLine();
while (s != null) {
  string[] v = s.Split(';');
  labs.Add(new Laboratorio {Id = int.Parse(v[0]), Descricao=v[1]});
  s = f.ReadLine();
}
f.Close();

foreach(Laboratorio lab in labs) Console.WriteLine(lab);
```

```
1 - Lab01
2 - Lab02
```

Arquivo XML

- Para escrever arquivos XML
  - Utilizar a classe StreamWriter para escrever o arquivo
  - A serialização usa a classe XmlSerializer do namespace System.Xml.Serialization
  - XmlSerializer não é genérica: o operador *typeof* informa a classe a ser serializada

```
List<Laboratorio> labs = new List<Laboratorio>();
labs.Add(new Laboratorio {Id = 1, Descricao = "Lab01"});
labs.Add(new Laboratorio {Id = 2, Descricao = "Lab02"});

XmlSerializer xml = new XmlSerializer(typeof(List<Laboratorio>));
StreamWriter f = new StreamWriter("Laboratorio.xml");
xml.Serialize(f, labs);
f.Close();
```

Lista de Laboratórios em XML

- Objetos serializados pela classe XmlSerializer

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfLaboratorio xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Laboratorio>
    <Id>1</Id>
    <Descricao>Lab01</Descricao>
  </Laboratorio>
  <Laboratorio>
    <Id>2</Id>
    <Descricao>Lab02</Descricao>
  </Laboratorio>
</ArrayOfLaboratorio>
```

Arquivo XML

- Para ler arquivos XML
  - Utilizar a classe StreamReader para ler o arquivo
  - De-serializar usando a classe XmlSerializer
  - XmlSerializer não é genérica: é necessário o *type-cast* após a de-serialização

```
XmlSerializer xml = new XmlSerializer(typeof(List<Laboratorio>));
StreamReader f = new StreamReader("Laboratorio.xml");
List<Laboratorio> labs = (List<Laboratorio>)xml.Deserialize(f);
f.Close();

foreach(Laboratorio lab in labs) Console.WriteLine(lab);
```

```
1 - Lab01
2 - Lab02
```

Arquivo Json

- Para escrever arquivos Json
  - Utilizar a classe StreamWriter ou File para escrever o arquivo
  - A serialização usa a classe JsonSerializer do namespace System.Text.Json
  - JsonSerializer é genérica: usa um parâmetro de tipo para a classe a ser serializada

```
List<Laboratorio> labs = new List<Laboratorio>();
labs.Add(new Laboratorio { Id = 1, Descricao = "Lab01" });
labs.Add(new Laboratorio { Id = 2, Descricao = "Lab02" });

string s = JsonSerializer.Serialize<List<Laboratorio>>(labs);
File.WriteAllText("Laboratorio.json", s);
```

```
[{"Id":1,"Descricao":"Lab01"},{"Id":2,"Descricao":"Lab02"}]
```

Arquivo Json

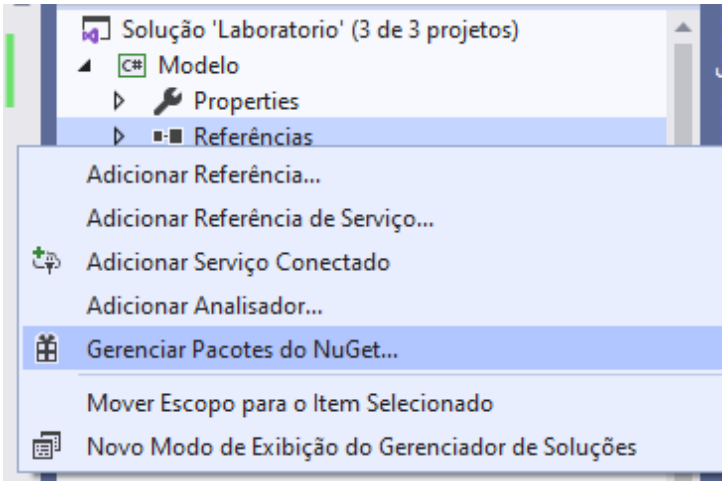
- Para ler arquivos Json
  - Utilizar a classe StreamReader ou File para ler o arquivo
  - De-serializar usando a classe JsonSerializer
  - JsonSerializer é genérica: não é necessário *type-cast*

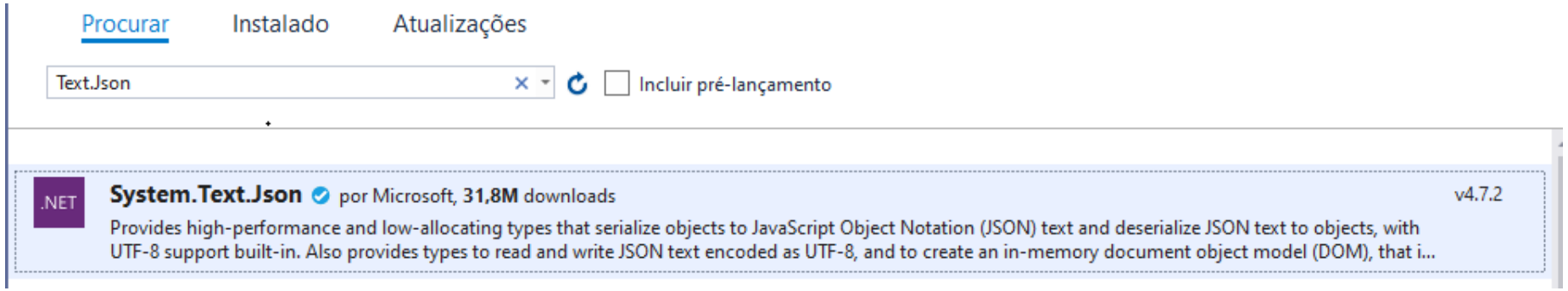
```
string s = File.ReadAllText("Laboratorio.json");
List<Laboratorio> labs = JsonSerializer.Deserialize<List<Laboratorio>>(s);

foreach(Laboratorio lab in labs) Console.WriteLine(lab);
```

```
1 - Lab01
2 - Lab02
```

Instalação do Pacote Json

- É possível que o pacote System.Text.Json necessite de instalação adicional





Referências

- Microsoft Visual C# 2010 – Passo a passo, John Sharp, Bookman, 2010
- Serialização no .NET
  - https://docs.microsoft.com/pt-br/dotnet/standard/serialization/
- StreamWriter Class
  - https://docs.microsoft.com/pt-br/dotnet/api/system.io.streamwriter?view=netcore-3.1
- StreamReader Class
  - https://docs.microsoft.com/pt-br/dotnet/api/system.io.streamreader?view=netcore-3.1
- File Class
  - https://docs.microsoft.com/pt-br/dotnet/api/system.io.file?view=netcore-3.1