

Data Science : Save and Load com JSON

Prof. Ovídio Francisco



Plano de aula

Persistência de dados em jogos

A Classe PlayerPrefs

Métodos

Exemplo de uso

Exercícios

Introdução

O contexto

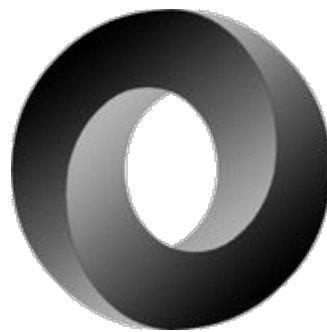
Com a popularização de diversas tecnologias de software e dispositivos eletrônicos, surgiu a necessidade de compartilhar dados entre plataformas diferentes.

Por exemplo, aplicativos mobile que compartilham dados com sites e sistemas escritos em linguagens de programação distintas precisam de um padrão para salvar, carregar e transmitir dados.

As soluções como arquivos de configuração não possuem um formato unificado. Já os sistemas gerenciadores de bancos de dados eram complexos demais por se tratar de uma tecnologia aplicada a grandes sistemas tradicionais. Assim, os desenvolvedores precisavam de uma forma simples, fácil e independente de linguagens e bibliotecas específicas.

Com isso, a linguagem JavaScript trouxe o JSON como solução para um formato de troca de dados entre softwares.

JSON



JSON (JavaScript Object Notation) é um formato de texto estruturado desenvolvido para representar objetos entre sistemas distintos.

- Independente de linguagem de programação
- Fácil leitura e escrita por humanos
- Fácil criação e extração de dados por máquinas
- Popular e amplamente utilizado
- Facilidade de registro em arquivos de texto

JSON

Um **objeto** representado em **Json** é delimitado por chaves “{” e “}” entre as quais contém seus **dados na forma chave-valor** e **outros objetos** separados por uma vírgula “,”.

Uma chave é um **identificador** de um **atributo** e deve ser seguido de um “:”. Veja o exemplo abaixo um jogador tem os atributos name, lives e health.

```
{  
  "playerName": "Sagan",  
  "lives": 5,  
  "health": 0.72  
}
```

** Há ainda uma forma menos utilizada que pode representar objetos como listas de valores.*

JSON

Abaixo está um exemplo onde um objeto é parte de outro.

```
{  
  "playerName": "Darwin",  
  "position": {  
    "x": -4.662805557250977,  
    "y": -2.941060781478882,  
    "z": 0.0  
  },  
  "lives": 2,  
  "score": 280,  
  "timeLeft": 86.18439483642578  
}
```

A classe JsonUtility

A classe **JsonUtility** fornece métodos estáticos para trabalhar com objetos na notação Json. São eles:

Método	Descrição
ToJson	Cria a representação Json de um objeto C#
FromJson	Cria um objeto C# a partir de sua representação Json
FromJsonOverwrite	Sobrescreve os dados de um objeto a partir de sua representação Json

JsonUtility - Exemplo ToJson

O método ToJson cria uma representação Json a partir dos atributos públicos de um dado objeto. Veja o exemplo adaptado da documentação da classe JsonUtility:

```
public class PlayerState : MonoBehaviour
{
    public string playerName;
    public int    lives;
    public float  health;

    public string SaveToString()
    {
        return JsonUtility.ToJson(this);
    }
}
```

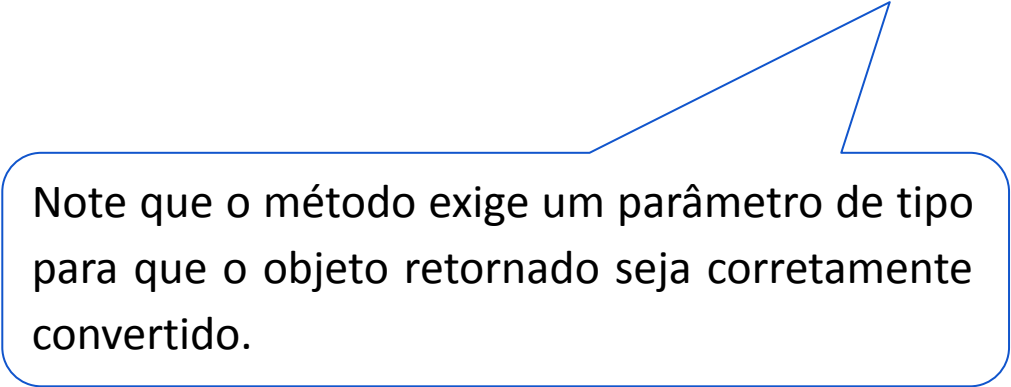
Gera como resultado algo como:

```
{"playerName":"Asdrubal","lives":3,"health":0.8}
```

JsonUtility - Exemplo FromJson

O método FromJson cria uma instância de um objeto a partir de sua representação Json. Veja o exemplo:

```
string json = "{\"playerName\":\"Nicola\",\"lives\":9,\"health\":0.4}";  
PlayerState ps = JsonUtility.FromJson<PlayerState>(json);
```



Note que o método exige um parâmetro de tipo para que o objeto retornado seja corretamente convertido.

JsonUtility - Exemplo FromJsonOverwrite

O método **FromJsonOverwrite** sobrescreve os dados de um objeto já instanciado. Esses dados serão lidos de uma representação **Json** e copiados para o objeto. Vale observar que o objeto **não será criado**, apenas seus dados serão sobrescritos. Veja o exemplo:

```
string json = "{\"playerName\":\"Ada Lovelace\",\"lives\":2,\"health\":0.95}";  
  
JsonUtility.FromJsonOverwrite(player_state, json);
```



Instância previamente criada.

Salvando na prática



Salvando dados em Json na prática

Nesse exemplo vamos partir de jogo simples do qual alguns dados serão salvos e depois restaurados.



Salvando dados em Json na prática

Em um ambiente Unity o desenvolvedor pode salvar dados do jogo seguindo os passos:

- Identificar os dados que deseja salvar
- Usar classes que concentram esses dados
- Criar uma representação Json do objeto
- Salvar resultado em um arquivo

Salvando dados em Json na prática

Veja a definição da classe `GameData` que armazena dados do jogo em atributos públicos.

```
public class GameData {  
    public Vector3 position;  
    public float timeLeft;  
}
```


Salvando dados em Json na prática

Veja também a classe SaveLoadJSON e o método SaveToJsonFile que atribui dados do game a uma instância de GameData. Em seguida, cria uma string contendo a representação json e a salva em um arquivo de texto.

```
public class SaveLoadJSON : MonoBehaviour
{
    public Player player;

    public void SaveToJsonFile() {
        GameData gd = new GameData();

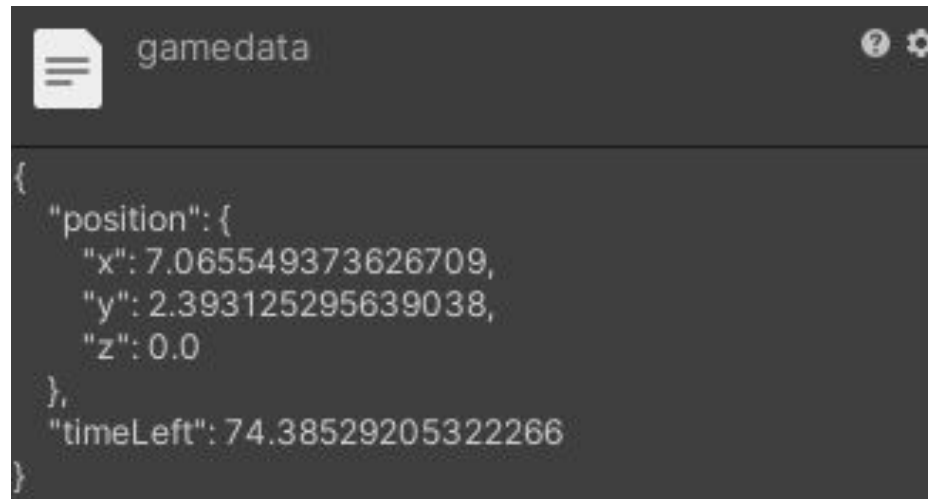
        gd.position = player.transform.position;
        gd.timeLeft = player.timeLeft;

        string json = JsonUtility.ToJson(gd, true);

        File.WriteAllText(Application.dataPath + "/gamedata.json", json);
    }
}
```

Salvando dados em Json na prática

Nesse exemplo o arquivo foi criado no diretório Assets do projeto, e pode ser facilmente acessado.

A screenshot of a code editor window. The title bar at the top shows a document icon, the filename 'gamedata', and two icons on the right (a question mark and a gear). The editor area has a dark background and displays a JSON object. The JSON structure is: an opening curly brace, followed by a quoted key 'position' with a colon and an opening curly brace, then three quoted keys 'x', 'y', and 'z' each followed by a colon and a numerical value (7.065549373626709, 2.393125295639038, and 0.0 respectively), then a closing curly brace, a comma, a quoted key 'timeLeft' with a colon and a numerical value 74.38529205322266, and finally a closing curly brace.

```
{  
  "position": {  
    "x": 7.065549373626709,  
    "y": 2.393125295639038,  
    "z": 0.0  
  },  
  "timeLeft": 74.38529205322266  
}
```

Resgatando na prática



Resgatando dados de um Json na prática

Em um ambiente Unity o desenvolvedor pode resgatar dados de jogo já encerrado, porém salvo, seguindo os passos:

- Carregar o conteúdo de um arquivo Json para um string
- Usar o método FromJson para criar um objeto com os dados salvos
- Atribuir os dados aos objetos do game

Resgatando dados de um Json na prática

Observe a implementação do método LoadFromJsonFile também da classe SaveLoadJson. Esse método carrega o arquivo gamedata.json e utiliza seu conteúdo para criar um objeto GameData. Com esses dados, o game é então configurado.

```
public void LoadFromJsonFile() {  
    string json = File.ReadAllText(Application.dataPath + "/gamedata.json");  
  
    GameData gd = JsonUtility.FromJson<GameData>(json);  
  
    player.transform.position = gd.position;  
    player.timeLeft = gd.timeLeft;  
}
```

Para saber mais...

<https://www.json.org/json-en.html>

<https://docs.unity3d.com/ScriptReference/JsonUtility.html>

<https://docs.unity3d.com/ScriptReference/JsonUtility.ToJson.html>

<https://docs.unity3d.com/ScriptReference/JsonUtility.FromJson.html>

<https://docs.unity3d.com/ScriptReference/JsonUtility.FromJsonOverwrite.html>

https://www.w3schools.com/whatis/whatis_json.asp

https://www.w3schools.com/js/js_json_intro.asp

Exercícios

Exercícios AC1

A partir do projeto inicial, fornecido pelo professor, crie as funcionalidades de salvar e carregar o jogo usando JSON.

Salve e carregue pelo menos 5 atributos do jogo.

Entrega individual pelo Canvas.

Serão considerados para nota:

- A corretude (deve funcionar como esperado)
- A completude (pelo menos 5 atributos)
- A estética e experiência do usuário.

Opcionalmente o aluno pode fazer melhorias, que serão consideradas para a nota.