OFPPT

Rapport Sur Fichier JSON

Réalisée par

Safae **E**louatyq

Manal **B**enzakour

Ilham **S**ahmoudi

Fatima ezzahraa **H**adrouch

Encadrée par

Mr HAQAY

Sommair

1) Introduction.....3

2) Concepts Clés de JSON.....3

3) Sérialisation et Désérialisation.....3

4) JSON en JavaScript.....4

5) JSON en Python.....4

6) Bonnes Pratiques.....5

7) Cas d'Utilisation.....5

8) Conclusion.....5

1) INTRODUCTION :

JSON (JavaScript Object Notation) est un format de données léger, largement utilisé pour échanger des informations entre les systèmes. Dans le cadre de la programmation orientée objet (POO), JSON joue un rôle crucial en permettant la sérialisation et la désérialisation des objets. Ce rapport explore l'intégration de JSON dans divers langages de programmation orientés objet à travers des exemples pratiques.

2) Concepts Clés de JSON

JSON est basé sur la syntaxe des objets JavaScript et se compose de Paires clé/valeur et de listes ordonnées de valeurs. Voici un exemple de structure JSON :

```
json
{
   "nom": "Jean",
    "âge": 30,
    "profession": "Développeur",
    "compétences": ["JavaScript", "HTML", "CSS"]
}
```

3) Sérialisation et Désérialisation

La sérialisation consiste à convertir un objet en une chaîne JSON, tandis que la désérialisation consiste à convertir une chaîne JSON en un objet.

4) JSON en JavaScript :

JavaScript offre une prise en charge native de JSON via les méthodes **JSON.stringify()** et **JSON.parse()**.

Exemple en JavaScript :

```
javascript
const personne = {
    nom: "Jean",
    âge: 30,
    profession: "Développeur"
};

const jsonString = JSON.stringify(personne);
console.log(jsonString);  // {"nom":"Jean","âge":30,"profession":"Développeur"}

const personneObj = JSON.parse(jsonString);
console.log(personneObj.nom);  // Jean
```

5) JSON en Python :

Python utilise le module json pour sérialiser et désérialiser des objets.

Exemple en Python :

```
python

import json

personne = {
    "nom": "Jean",
    "âge": 30,
    "profession": "Développeur"
}

json_string = json.dumps(personne)
print(json_string)  # {"nom": "Jean", "âge": 30, "profession": "Développeur"}

personne_obj = json.loads(json_string)
print(personne_obj['nom'])  # Jean
```

6) Bonnes Pratiques :

Valider les données JSON avant la désérialisation.

Implémenter une gestion des erreurs robuste.

Garder les objets simples pour faciliter la sérialisation et la désérialisation.

7) Cas d'Utilisation :

- APIs REST
- Stockage de Configuration
- Persistant Data Storage

8) Conclusion

L'intégration de JSON dans la POO permet *une manipulation* efficace des données structurées. JSON, en raison de sa simplicité et de son interopérabilité, est largement utilisé dans de nombreux contextes, renforçant son utilité dans la communication entre systèmes et les stockage de données.