SAÉ 1.05 : TRAITER DES DONNÉES

Martin Pépin

2 décembre 2024
BUT RT 1A — Ifs — SAÉ1.05

SAE1.05

La SAE1.05 propose de réaliser un petit projet de développement via différentes étapes pour centraliser des données et les mettre en forme.

Elle mobilise les compétences acquises de :

- R106 Architecture des systèmes numériques et informatiques
- R107 Fondamentaux de la programmation
- R108 Bases des systèmes d'exploitation
- R109 Introduction aux technologies Web
- R110 Anglais technique 1
- R111 Expression-Culture-Communication Professionnelles : Introduction à la communication et au savoir-être professionnels
- R112 Projet Personnel et Professionnel
- R115 Gestion de projet 1 : Maîtriser les bases de l'organisation du travail

Rappel des coefficients

Coefficients du Tronc Commun au S1					
		RT1 - Administrer	RT2 - Connecter	RT3 - Programmer	Total toutes compétences
SAÉ1.01	Se sensibiliser à l'hygiène informatique et à la...	10			
SAÉ1.02	S'initier aux réseaux informatiques	31			
SAÉ1.03	Découvrir un dispositif de transmission		36		
SAÉ1.04	Se présenter sur Internet			8	
SAÉ1.05	Traiter des données			35	
SAÉ1.PORTFOLIO	Portfolio	0	0	0	
R1.01	Initiation aux réseaux informatiques	13	4	4	
R1.02	Principes et architecture des réseaux	12			
R1.03	Réseaux locaux et équipements actifs	7	2	2	
R1.04	Fondamentaux des systèmes électroniques	8	8		
R1.05	Supports de transmission pour les réseaux		6		
R1.06	Architecture des systèmes numériques et...	5		5	
R1.07	Fondamentaux de la programmation			19	
R1.08	Bases des systèmes d'exploitation	6		6	
R1.09	Introduction aux technologies Web			4	
R1.10	Anglais technique 1	5	5	5	
R1.11	Expression-Culture-Communication...	4	5	5	
R1.12	Projet Personnel et Professionnel	2	2	2	
R1.13	Mathématiques du signal	5	9		
R1.14	Mathématiques des transmissions	5	9		
R1.15	Gestion de projet 1 : Maîtriser les bases de...		3	3	
	Total sur les SAEs	**41**	**36**	**43**	120
	Total sur les ressources	**72**	**53**	**55**	180
	Total sur les SAEs et les ressources	**113**	**89**	**98**	300

Problématique professionnelle (extrait PN)

Le professionnel R&T est régulièrement amené à **traiter des données** provenant du système d'information de l'entreprise pour ses besoins personnels ou ceux de ses collaborateurs. Ces données peuvent par exemple être liées à l'infrastructure de son réseau (état des équipements, des machines) ou aux utilisateurs.

Généralement **obtenues sous forme brutes**, elles sont ensuite **traitées** avec des objectifs très variés (nettoyage des données, extraction d'informations comptables, archivage, ...) pour être **réutilisées** à d'autres fins ou être **présentées** dans des vues synthétiques.

Ces traitements peuvent être récurrents (mensualisation de bilan, sauvegarde de données périodique, ...) et **gagnent à être automatisés**. Le professionnel R&T doit donc développer des scripts ou des programmes pour gérer de façon efficace le traitement de ces données.

En pratique...

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes...

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

1. Récolter des données via une API en ligne (PokeAPI)

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

1. Récolter des données via une API en ligne (PokeAPI)
2. Mettre en forme les données selon les informations désirées

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

1. Récolter des données via une API en ligne (PokeAPI)
2. Mettre en forme les données selon les informations désirées
3. Effectuer des traitements statistiques

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

1. Récolter des données via une API en ligne (PokeAPI)
2. Mettre en forme les données selon les informations désirées
3. Effectuer des traitements statistiques
4. Exporter les stats via un format particulier (Markdown)

En pratique…

Avec vos compétences, vous devez être capables de :

- créer des « petits » programmes…
- pour automatiser le traitement de données…
- et produire des statistiques.

L'objectif est dans notre cas le suivant :

1. Récolter des données via une API en ligne (PokeAPI)
2. Mettre en forme les données selon les informations désirées
3. Effectuer des traitements statistiques
4. Exporter les stats via un format particulier (Markdown)

Avec du temps et de l'expérience, on peut aboutir à ce genre de projet (très réussi) de collecte et mise en forme de données : `https://tif.hair/`

Organisation et notation

- 1h de CM

- Pour les initiaux :
  - 4h de TP pour vous lancer sur le projet,
  - 12h en autonomie pour réaliser le travail demandé dans le cahier des charges.

- Pour les alternant·es :
  - 7h30 de TP pour vous accompagner sur le projet,
  - 2h30 en autonomie pour réaliser le travail demandé dans le cahier des charges.

  → Moins d'heures donc moins de travail attendu.

- Note finale = sur le projet

Plan

Prérequis

Vous avez déjà un bagage technique en Python.

Prérequis

Vous avez déjà un bagage technique en Python.

(Vous connaissez déjà les listes, dictionnaires, etc.)

Prérequis

Vous avez déjà un bagage technique en Python.

(Vous connaissez déjà les listes, dictionnaires, etc.)

Mais avant de partir dans le vif du sujet, nous avons besoin d'introduire quelques pré-requis techniques supplémentaires :

Prérequis

Vous avez déjà un bagage technique en Python.

(Vous connaissez déjà les listes, dictionnaires, etc.)

Mais avant de partir dans le vif du sujet, nous avons besoin d'introduire quelques pré-requis techniques supplémentaires :

- interface de programmation (API)
- modèle de données JSON et XML
- modèle de communication RESTful

API

API (*Application Programming Interface*) : ensemble d'outils qui permet à deux applications de communiquer entre elles.

API

API (*Application Programming Interface*) : ensemble d'outils qui permet à deux applications de communiquer entre elles.

Une API fournit une manière d'accéder aux données d'une application :

- comment interroger l'application ?
- quels sont les paramètres requis et optionnels ?
- quel est le format des données renvoyées par le serveur ?

API

API (*Application Programming Interface*) : ensemble d'outils qui permet à deux applications de communiquer entre elles.

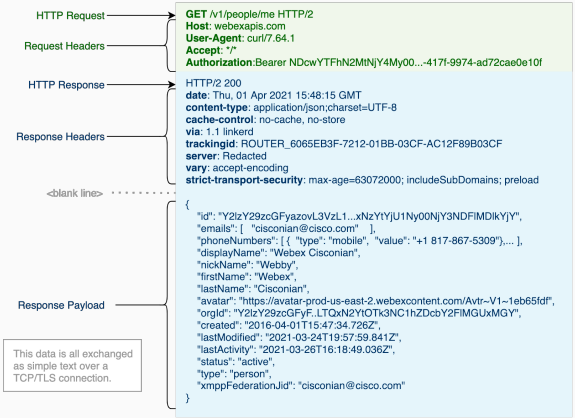
Une API fournit une manière d'accéder aux données d'une application :

- comment interroger l'application ?
- quels sont les paramètres requis et optionnels ?
- quel est le format des données renvoyées par le serveur ?

Exemple d'utilisation d'une API : `http://api.steampowered.com/ISteamNews/GetNewsForApp/v0002/?appid=730&count=3&maxlength=300&format=json`

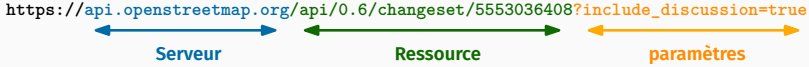
API REST

Les **API REST** (*representational state transfer*) sont basées sur le protocole HTTP et ses différentes méthodes afin d'interagir avec un serveur et manipuler ses données :



API REST

Une requête REST a plusieurs paramètres, certains optionnels :



API REST

Une requête REST a plusieurs paramètres, certains optionnels :

`https://api.openstreetmap.org/api/0.6/changeset/5553036408?include_discussion=true`

On peut utiliser plusieurs méthodes, et les requêtes produisent différents codes de retour

HTTP Verb	Typical Action (CRUD)
`POST`	Create
`GET`	Read
`PUT`	Update
`PATCH`	Update
`DELETE`	Delete

Status Code	Status Message	Meaning
200	OK	All looks good
201	Created	New resource created
400	Bad Request	Request was invalid
401	Unauthorized	Authentication missing or incorrect
403	Forbidden	Request was understood but not allowed
404	Not Found	Resource not found
500	Internal Server Error	Something wrong with the server
503	Service Unavailable	Server is unable to complete request

API REST

Une requête REST a plusieurs paramètres, certains optionnels :

`https://api.openstreetmap.org/api/0.6/changeset/5553036408?include_discussion=true`

Serveur — **Ressource** — **paramètres**

On peut utiliser plusieurs méthodes, et les requêtes produisent différents codes de retour

HTTP Verb	Typical Action (CRUD)
`POST`	Create
`GET`	Read
`PUT`	Update
`PATCH`	Update
`DELETE`	Delete

Status Code	Status Message	Meaning
200	OK	All looks good
201	Created	New resource created
400	Bad Request	Request was invalid
401	Unauthorized	Authentication missing or incorrect
403	Forbidden	Request was understood but not allowed
404	Not Found	Resource not found
500	Internal Server Error	Something wrong with the server
503	Service Unavailable	Server is unable to complete request

Dans certains cas, les API REST demandent une clé d'authentification

Format de données

En utilisant une API, un programme peut interagir avec un autre programme.
Cependant, le format des données renvoyé n'est pas toujours standard !

Format de données

En utilisant une API, un programme peut interagir avec un autre programme. Cependant, le format des données renvoyé n'est pas toujours standard !

Prenons par exemple le résultat d'une commande en CLI Cisco pour afficher la liste des interfaces réseau :

```
switch# show the int bri

--------------------------------------------------------------------------------
Ethernet      VLAN    Type Mode   Status  Reason                   Speed     Port
Interface                                                                    Ch #
--------------------------------------------------------------------------------
Eth1/1        1       eth  fabric down    SFP not inserted           10G(D) --
```

Format de données

En utilisant une API, un programme peut interagir avec un autre programme. Cependant, le format des données renvoyé n'est pas toujours standard !

Prenons par exemple le résultat d'une commande en CLI Cisco pour afficher la liste des interfaces réseau :

```
switch# show the int bri

--------------------------------------------------------------------------------
Ethernet      VLAN    Type Mode   Status  Reason                   Speed     Port
Interface                                                                    Ch #
--------------------------------------------------------------------------------
Eth1/1        1       eth  fabric down    SFP not inserted           10G(D) --
```

Le format renvoyé par l'IOS Cisco est en texte clair : conçu pour être lu par un humain. Ce format n'est pas du tout adapté aux outils de programmation ! Où sont les mots-clés ? Où est l'information utile ?

Format de données : XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:if_manager">
 <nf:data>
  <show>
   <interface>
    <__XML__INTF_ifeth_brf>
     <__XML__PARAM_value>
      <__XML__INTF_output>Ethernet1/1-3</__XML__INTF_output>
     </__XML__PARAM_value>
     <__XML__OPT_Cmd_show_interface_if_eth_brief___readonly__>
      <__readonly__>
       <TABLE_interface>
        <ROW_interface>
         <interface>Ethernet1/1</interface>
         <vlan>1</vlan>
         <type>eth</type>
         <portmode>Fabric</portmode>
         <state>down</state>
         <state_rsn_desc>SFP not inserted</state_rsn_desc>
         <speed>10G</speed>
         <ratemode>D</ratemode>
        </ROW_interface>
       </TABLE_interface>
      </__readonly__>
     </__XML__OPT_Cmd_show_interface_if_eth_brief___readonly__>
    </__XML__INTF_ifeth_brf>
   </interface>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

XML (*eXtensible Markup Language*) est un format de données organisé sous forme de balises. Il est fortement structuré, ce qui évite les malformations et qui permet de vérifier l'intégrité des données.

Format de données : XML

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<nf:rpc-reply xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://www.cisco.com/nxos:1.0:if_manager">
 <nf:data>
  <show>
   <interface>
    <__XML__INTF_ifeth_brf>
     <__XML__PARAM_value>
      <__XML__INTF_output>Ethernet1/1-3</__XML__INTF_output>
     </__XML__PARAM_value>
     <__XML__OPT_Cmd_show_interface_if_eth_brief___readonly__>
      <__readonly__>
       <TABLE_interface>
        <ROW_interface>
         <interface>Ethernet1/1</interface>
         <vlan>1</vlan>
         <type>eth</type>
         <portmode>Fabric</portmode>
         <state>down</state>
         <state_rsn_desc>SFP not inserted</state_rsn_desc>
         <speed>10G</speed>
         <ratemode>D</ratemode>
        </ROW_interface>
       </TABLE_interface>
      </__readonly__>
     </__XML__OPT_Cmd_show_interface_if_eth_brief___readonly__>
    </__XML__INTF_ifeth_brf>
   </interface>
  </show>
 </nf:data>
</nf:rpc-reply>
]]>]]>
```

XML (*eXtensible Markup Language*) est un format de données organisé sous forme de balises. Il est fortement structuré, ce qui évite les malformations et qui permet de vérifier l'intégrité des données. HTML est dérivé de XML, c'est assez similaire !

Format de données : JSON


```
{
    "interfaces": [
        {
            "interface": "Ethernet1/1",
            "vlan": "1",
            "type": "eth",
            "portmode": "fabric",
            "state": "down",
            "state_rsn_desc": "SFP not inserted",
            "speed": "10G",
            "ratemode": "D"
        }
    ]
}
```


JSON (*JavaScript Object Notation*) est un format de données alternatif où les données sont moins structurées, mais où la lisibilité humaine est beaucoup plus accessible.

Exploitation d'une API

Dans le cadre de cette SAÉ, nous travaillerons avec une API REST.
Il existe plusieurs façon d'interagir avec :

Exploitation d'une API

Dans le cadre de cette SAÉ, nous travaillerons avec une API REST.
Il existe plusieurs façon d'interagir avec :

- Requêtes HTTP directement via un navigateur :
  - ✓ pratique pour tester / expérimenter,
  - ✗ on ne récupère pas les données pour les traiter plus tard ;

Exploitation d'une API

Dans le cadre de cette SAÉ, nous travaillerons avec une API REST.
Il existe plusieurs façon d'interagir avec :

- Requêtes HTTP directement via un navigateur :
  - ✓ pratique pour tester / expérimenter,
  - ✗ on ne récupère pas les données pour les traiter plus tard ;

- Bibliothèque `requests` en Python pour faire des appels HTTP :
  - ✓ on peut récupérer les données,
  - ✓ conversion automatique en objets Pythons,
  - ✓ on peut automatiser.

Bibliothèque `requests`

Exemple d'utilisation :

```
import requests


api_url = "https ://jsonplaceholder.typicode.com/todos/1"
response = requests.get(api_url)
donnees = response.json()
```

Plan

Markdown : généralités

Markdown est un langage de balisage léger qui permet de mettre en forme des documents texte avec une syntaxe facile à lire et à écrire. Un document au format `.md` est très humain-compatible.

Markdown : généralités

Markdown est un langage de balisage léger qui permet de mettre en forme des documents texte avec une syntaxe facile à lire et à écrire. Un document au format `.md` est très humain-compatible.

On peut « compiler » un fichier Markdown en PDF ou en page HTML.

Markdown : généralités

Markdown est un langage de balisage léger qui permet de mettre en forme des documents texte avec une syntaxe facile à lire et à écrire. Un document au format `.md` est très humain-compatible.

On peut « compiler » un fichier Markdown en PDF ou en page HTML.

Créé en 2004, il est notamment beaucoup utilisé pour les blogs, forums simples, et surtout les outils de développement collaboratifs et autres fichiers README.

Markdown : généralités

- Prenons par exemple l'écriture de cette phrase en HTML :
  `<p> Téma la taille du <strong> rat </strong> </p>`

Markdown : généralités

- Prenons par exemple l'écriture de cette phrase en HTML :
  ```
  <p> Téma la taille du <strong> rat </strong> </p>
  ```

- On peut l'écrire de la même manière en LaTeX :
  ```
  Téma la taille du \textbf{rat}
  ```

Markdown : généralités

- Prenons par exemple l'écriture de cette phrase en HTML :
  ```
  <p> Téma la taille du <strong> rat </strong> </p>
  ```

- On peut l'écrire de la même manière en LaTeX :
  ```
  Téma la taille du \textbf{rat}
  ```

  Dans les deux cas précédents, les documents vont bien s'afficher comme voulu mais relecture peut être pénible.

Markdown : généralités

- Prenons par exemple l'écriture de cette phrase en HTML :
  ```
  <p> Téma la taille du <strong> rat </strong> </p>
  ```

- On peut l'écrire de la même manière en LaTeX :
  ```
  Téma la taille du \textbf{rat}
  ```

  Dans les deux cas précédents, les documents vont bien s'afficher comme voulu mais relecture peut être pénible.

- En Markdown, c'est bien plus simple :
  ```
  Téma la taille du **rat**
  ```

Markdown : syntaxe

Cette page peut vous servir de rappel de la syntaxe de Markdown :

https://markdown-guide.readthedocs.io/en/latest/basics.html

Les essentiels :

Markdown : syntaxe

Cette page peut vous servir de rappel de la syntaxe de Markdown :

`https://markdown-guide.readthedocs.io/en/latest/basics.html`

Les essentiels :

- Deux sauts de lignes pour séparer les paragraphes

Markdown : syntaxe

Cette page peut vous servir de rappel de la syntaxe de Markdown :

https://markdown-guide.readthedocs.io/en/latest/basics.html

Les essentiels :

- Deux sauts de lignes pour séparer les paragraphes
- Le gras et l'italique se gèrent avec des étoiles :

```
*Texte en italique*
**Texte en gras**
*** Gras et italique***
```

Markdown : syntaxe

Cette page peut vous servir de rappel de la syntaxe de Markdown :

https://markdown-guide.readthedocs.io/en/latest/basics.html

Les essentiels :

- Deux sauts de lignes pour séparer les paragraphes
- Le gras et l'italique se gèrent avec des étoiles :

```
*Texte en italique*
**Texte en gras**
*** Gras et italique***
```

- Les titres sont définis avec des dièses :

```
# titre principal (h1)
## sous titre (h2)
### sous-sous titre (h3)
#### ...
```

Markdown : syntaxe

- Listes ordonnées et non ordonnées (pour les sous-listes, indentez de 4 espaces) :

```
1. Premier élément
2. Deuxième élément
    - sous-élément
    - autre sous-élément
```

- Pour insérer une image :

```
![alt text](https ://i.imgur.com/Ssns07U.jpeg)
![](./images/bruh.png)
```

- Et plein d'autres outils assez faciles à aborder... Pour se simplifier la vie, on peut tester avec un outil en ligne :

  `https://markdown-editor.github.io/`

Markdown : en Python

Si Markdown est un langage simple, il faut tout de même utiliser un compilateur pour le transformer dans un autre format. En Python, on peut utiliser la bibliothèque du même nom :

```
import markdown

with open('file.md', 'r') as f :
    text = f.read()

html = markdown.markdown(text)

with open('file.html', 'w') as f :
    f.write(html)
```

Plan

Plan

Conclusion

Quelques nouvelles notions :

- Les API, et plus particulièrement les API REST
- Les formats de donnée XML et JSON
- Comment interagir avec une API

Conclusion

Quelques nouvelles notions :

- Les API, et plus particulièrement les API REST
- Les formats de donnée XML et JSON
- Comment interagir avec une API

Donc en bref,

- pas beaucoup de nouveautés, du moins rien de très compliqué

Conclusion

Quelques nouvelles notions :

- Les API, et plus particulièrement les API REST
- Les formats de donnée XML et JSON
- Comment interagir avec une API

Donc en bref,

- pas beaucoup de nouveautés, du moins rien de très compliqué
- le but de cette SAÉ est d'aller plus loin que dans les ressources classiques et de réaliser un vrai projet de programmation.

Conclusion

Quelques nouvelles notions :

- Les API, et plus particulièrement les API REST
- Les formats de donnée XML et JSON
- Comment interagir avec une API

Donc en bref,

- pas beaucoup de nouveautés, du moins rien de très compliqué
- le but de cette SAÉ est d'aller plus loin que dans les ressources classiques et de réaliser un vrai projet de programmation.

À vous de jouer !