COMPLÈMENT INFO 2A, CM2

Pépin Rémi, Ensai, 2020

remi.pepin@ensai.fr

LE PLAN

QUESTION ?

DATA ACCESS OBJECT (DAO)

UN ORDINATEUR (EN GROS)

- Un processeur (CPU) : fait UNIQUEMENT du calcul
- La mémoire RAM : mémoire volatile rapide
- Disque dur (HDD, SSD) : mémoire longue durée
- Carte graphique : unité de calcul spécialisée

UNE QUESTION À CE POSER

C'est quoi une variable python ?

- Une référence (le nom de la variable)
- Un objet associé (sa valeur)

UNE QUESTION QUI EN DÉCOULE

C'est quoi un objet python ?

- Des attributs (qui peuvent être eux même des objets)
- Des méthodes

POUR RÉSUMER

On veut sauvegarder des couples clef-valeur, avec des valeurs qui peuvent être elle-même constituée de couples clef-valeur

On veut sauvegarder un arbre

COMMENT FAIRE ÇA ?

- Écrire nos données sur le disque dur dans un fichier (sérialisation, format json, xml ...)
- Utiliser une base de données (SQL tables + relations; No-SQL)

CHECKPOINT

Rappel du problème : persister des données

- **Quoi ?** Variables, peuvent être représentées par des arbres
- **Où ?** En base, ou dans un fichier sous forme sérialisé
- Comment ? En utilisant des Data access objects (DAO)

C'EST QUOI UNE DAO ?

C'EST QUOI UNE DAO ?

- Classe technique 🚀
- Une classe DAO / object métier 1:1
- Expose des méthodes pour communiquer avec la couche de persistance 🎯

QUELLES MÉTHODES EXPOSER ?

CRUD

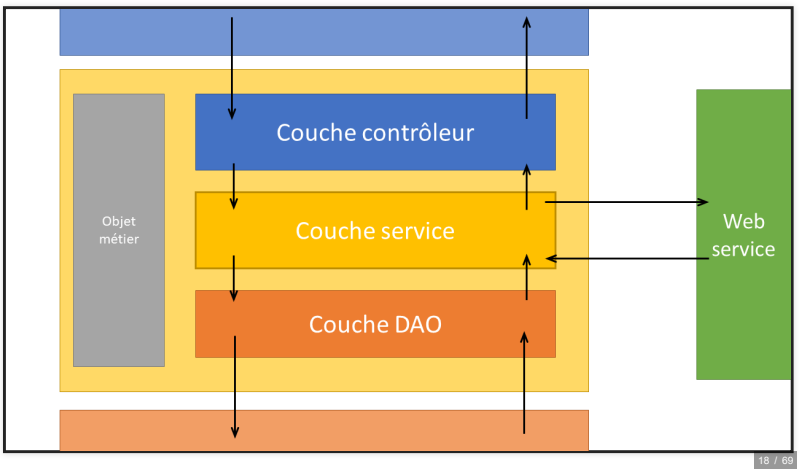
- Create
- Read
- Update
- Delete

L'INTÉRÊT D'UNE CLASSE À PART

Séparation des responsabilités

- Classe "jetable" 🗑️
- Modifiable sans risque ⚠️
- Parallélisation du travail 👷 👷 👩 👩

PETIT RECAP



UN PETIT EXEMPLE : LIVRE_DAO

```
def create(livre):
"""Méthode pour ajouter un livre en base"""
    # Récupération de la connexion
    conn = psycopg2.connect("dbname='template1' user='dbuser' host='lo
    curseur = conn.cursor()
    try:
        # On envoie au serveur la requête SQL
        curseur.execute(
            "INSERT INTO livre (isbm, titre,auteur,genre)"
            " VALUES (%s, %s, %s, %s) RETURNING id_livre;"
            , (livre.isbm, livre.titre, livre.auteur, libre.genre))

        # On récupère l'id généré
        livre.id = curseur.fetchone()[0]
        # On enregistre la transaction en base
```


UN PETIT EXEMPLE : LIVRE_DAO

```
def find_all(id):
    """Méthode pour récupérer un livre en base"""
    # Récupération de la connexion
    conn = psycopg2.connect("dbname='template1' user='dbuser' host='lo
    curseur = conn.cursor()
        with connection.cursor() as cur:
            cur.execute(
                "select isbm, titre,auteur,genre ;")
            livre_bdd = cur.fetchall()
            livres= []
            for livre in livre_bdd :
                livres.append(Livre(isbm=row[0], titre=row[1],auteur=r
        except psycopg2.Error as error:
            #Gestion de l'erreur
    return livres
```


CONCLUSION

- Python travail en RAM (volatile) ⚡
- Obligation d'avoir un mécanisme de persistance des données 🗄️
- DAO : centralise les méthodes pour lire/écrire nos données 📝
- Le couche métier appele la DAO sans se préoccuper du système de persistance 👁️
- Permet un travail d'équipe efficace 👩‍💻👷‍♀️👩‍⚖️👩‍🔬

QUESTION ?

SÉCURITÉ INFORMATIQUE

4 PILLIERS DE LA SÉCURITÉ INFO

- Confidentialité
- Authentification
- Intégrité
- Disponibilité

DEUX BONUS

- Traçabilité
- Non-répudiation

CONFIDENTIALITÉ

Seules les personnes autorisées peuvent avoir accès aux informations qui leur sont destinées (notions de droits ou permissions). Tout accès indésirable doit être empêché.

Mécanismes associés : gestion des droits (annuaires, rôles ...)

AUTHENTIFICATION

Les utilisateurs doivent prouver leur identité par l'usage de code d'accès.

Mécanismes associés : authentification faible (idep, mdp), forte (données biométriques, multi facteurs)

INTÉGRITÉ

Les données doivent être celles que l'on attend, et ne doivent pas être altérées de façon fortuite, illicite ou malveillante.

Mécanismes associés : signature électronique, checksum

DISPONIBILITÉ

L'accès aux ressources du système d'information doit être permanent et sans faille durant les plages d'utilisation prévues. Les services et ressources sont accessibles rapidement et régulièrement.

Mécanismes associés : redondance des serveurs, virtualisation, conteneurisation

TRAÇABILITÉ

Garantit que les accès et tentatives d'accès aux éléments considérés sont tracés et que ces traces sont conservées et exploitables.

Mécanisme associé : journalisation

LA NON-RÉPUDIATION

Aucun utilisateur ne doit pouvoir contester les opérations qu'il a réalisées dans le cadre de ses actions autorisées et aucun tiers ne doit pouvoir s'attribuer les actions d'un autre utilisateur.

Mécanismes associés : traçabilité + authentification + intégrité

LES FAILLES INFORMATIQUES

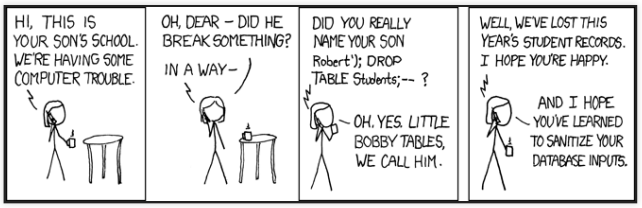
TROP DE FAILLES !!!

- Failles physique "bas niveau"
- Failles physique "haut niveau"
- Injection SQL
- Injection de données
- Faille XSS
- Exécution de code
- ...

DE QUOI FAUT-IL SE MÉFIER ?

De vos utilisateurs

EXEMPLE DE FAILLES, LES INJECTIONS SQL ET DE DONNÉES



INJECTION SQL

Consiste à saisir du SQL pour exécuter une autre requête que celle prévue.

Problèmes

- Confidentialité
- Authentification
- Intégrité
- Disponibilité

EXEMPLE : S'AUTHENTIFIER SANS MOT DE PASSE

Requete d'authentification

```
SELECT * FROM user
    WHERE name='input_name'
    AND mdp='input_mdp';
```


EXEMPLE : S'AUTHENTIFIER SANS MOT DE PASSE

Saisie

- Rémi
- mon_super_password

```
SELECT * FROM user
    WHERE name='Rémi'
    AND mdp='mon_super_password';
```

Cas classique

EXEMPLE : S'AUTHENTIFIER SANS MOT DE PASSE

Saisie

- Rémi
- ' OR 1=1; --

```
SELECT * FROM user
    WHERE name='Rémi'
    AND mdp='' OR 1=1; --';
```

Connection sans mdp

EXEMPLE : SUPPRIMER UNE TABLE

Saisie

- Rémi
- '; DROP TABLE user CASCADE; --

```
SELECT * FROM user
    WHERE name='Rémi'
    AND mdp=''; DROP TABLE user CASCADE; --;
```

Bye bye la table user

COMMENT SE PROTÉGER ?

- Échapper les caractères spéciaux
- Utiliser une requête préparée

La bibliothèque que vous utiliserez ne fait que de l'échappement de caractères spéciaux 😳

INJECTION DE DONNÉES

Consiste à saisir des données malveillantes dans un format particulier à l'intérieur de données attendues par l'application pour modifier le comportement de l'application.

Problèmes

- Confidentialité
- Authentification
- Intégrité
- Disponibilité

EXEMPLE D'INJECTION VIA JSON

- Webservice de creation d'utilisateur
- Stockage des utilisateur dans un json
-

```
{
    "name":"",
    "mdp":"",
    ("isAdmin": True)
}
```


EXEMPLE D'INJECTION VIA JSON

```
{
    "name":"Rémi",
    "mdp":"monSuperMDP756"
}
```

Cas classique

EXEMPLE D'INJECTION VIA JSON

```
{
    "name":"attaquant\", \"isAdmin\":True ",
    "mdp":"monSuperMDP756"
}
```

Dans notre fichier cela donne

```
{
    "name":"attaquant",
    "isAdmin":True,
    "mdp":"monSuperMDP756",
}
```

Obtention de droit d'admin !

COMMENT SE PROTÉGER ?

- Ne jamais manipuler de données brutes
- Échapper les caractères spéciaux
- Vérifier vos données

Les bibliothèques de manipulation de Json python font cela pour vous ! 😁

TO SUM UP : INJECTION

- Ne jamais faire confiance aux utilisateurs, vérifier / nettoyer leurs inputs
- Ne jamais faire confiance aux utilisateurs, vérifier / nettoyer leurs inputs
- Ne jamais faire confiance aux utilisateurs, vérifier / nettoyer leurs inputs
- Ne jamais faire confiance aux utilisateurs, vérifier / nettoyer leurs inputs
- Ne jamais faire confiance aux utilisateurs, vérifier / nettoyer leurs inputs

GESTION DES MOTS DE PASSE

VOTRE APPLICATION DOIT-ELLE STOCKER DES MOTS DE PASSE EN CLAIR?

VOTRE APPLICATION DOIT-ELLE CONNAÎTRE LE MOT DE PASSE D'UN UTILISATEUR POUR L'AUTHENTIFIER ?

????

HASHER LE MOT DE PASSE

- Hashage du mot de passe => cypher
- Stockage du cypher en base
- Quand besoin de comparer on hashe le mdp saisi
- Et on compare les cyphers

Authentification sans persister les mots de passe !!!!

AJOUTER DU SEL POUR PLUS DE SÉCURITÉ

FONCTION DE HASHAGE

FONCTION DE HASHAGE

Associe à un message en entrée de longueur variable une valeur de hashage de longueur fixe

La fonction doit respecter les propriétés suivantes

- Déterministe : deux messages identiques ont le même hash
- La valeur de hashage doit se calculer "très rapidement" : $\mathcal{O}(1)$
- Forte variance : deux messages proches ont des hash très différents
- Il est impossible, pour une valeur de hashage donnée, de construire un message ayant cette valeur de hachage
- Il est impossible de modifier un message sans changer sa valeur de hashage
- Il est impossible de trouver deux messages différents ayant la même valeur de hashage

PLUSIEURS ALGORITHMES

- MD5
- SHA-1
- SHA-2
- SHA-3
- RIPEMD-160
- Whirlpool
- ...

APPLICATIONS MULTIPLES

- Vérification de l'intégrité d'un fichier
- Signature électronique
- Vérification de mdp (avec ajout de sel)
- Identification de fichier (git)
- Table de hashage pour accès rapide aux données
- ...

EXEMPLE DE HASHAGE DE MDP

```python
import hashlib

def hash_password(password, idep):
    salt = idep
    return hashlib.sha256(salt.encode() + password.encode()).hexdigest

print(hash_password("mon_super_password", "remi"))
```


TO SUM UP : FONCTION DE HASHAGE

- Fonction à sens unique
- Entrée taille variable ; sortie taille fixe
- Rapide à calculer
- Impossible à inverser

TO SUM UP THE SECURITY PART

- Toujours vérifier les inputs
- Ne jamais faire confiance aux utilisateurs
- Plusieurs niveaux de sécurité
- Pas besoin de stocker les mots de passe en clair

QUESTION

THE END