Ingénierie Logicielle
Pattern DAO

Erick STATTNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-ag.fr

www.erickstattner.com

Description de l'enseignement

Objectifs pédagogiques

- Se familiariser avec les bonnes pratiques de développement
- En particulier avec l'accès aux données depuis un programme
- Mettre en place des applications qui utilisent le Pattern DAO

Organisation:

- 8h: CM et TP

Contexte

Dans la plupart des applications

- Des données sont manipulées à partir
  - De fichiers
  - De services WEB
  - De base de données
- Les objets manipulés sont souvent liés aux objets stockés
- On fait correspondre les objets aux données
- Les attributs de la table correspondent aux attributs de l'objet


```
public class Membre {
    private int identifiant;
    private String nom;
    private String prenom;

    public void modifierNom(String t){
        nom = t;
        //Modifier aussi dans la BD
    }
}
```

Contexte

Problèmes soulevés ?

Contexte

Problèmes soulevés ?

- Accès aux données dispersé
- Accès aux données confondu avec les classes "métier"
- En cas de changement du mode de stockage, recoder toutes les méthodes

Solution

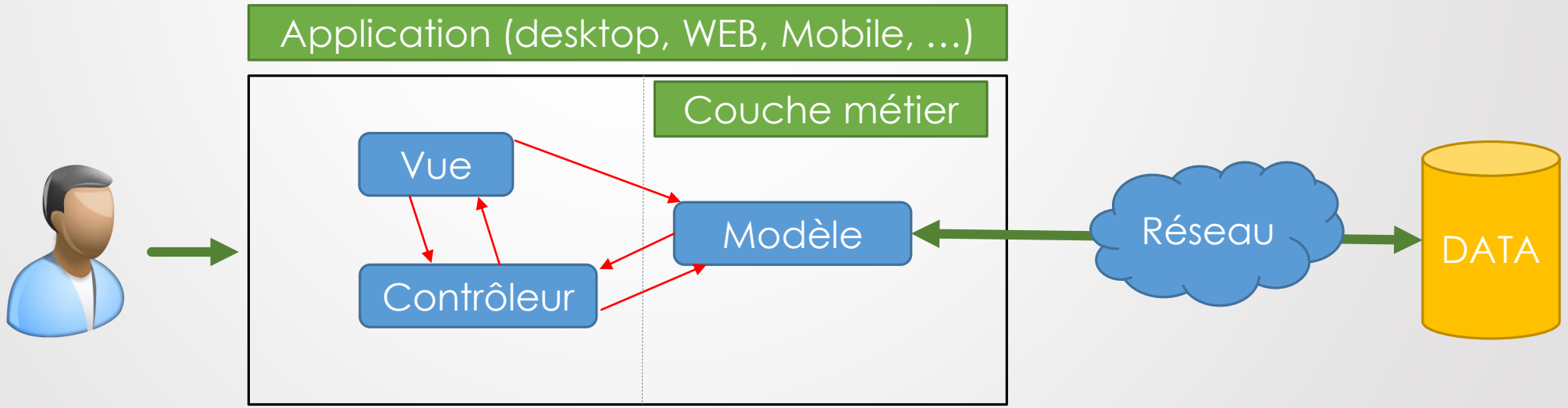
- Regrouper les accès aux données dans des classes à part
- S'abstraire de la façon dont les données sont stockées
  - Le changement du mode de stockage ne remet pas en cause le reste de l'application
- **Le Pattern DAO (Data Access Object)**

Sommaire

I. Introduction

Une application est découpée en couche

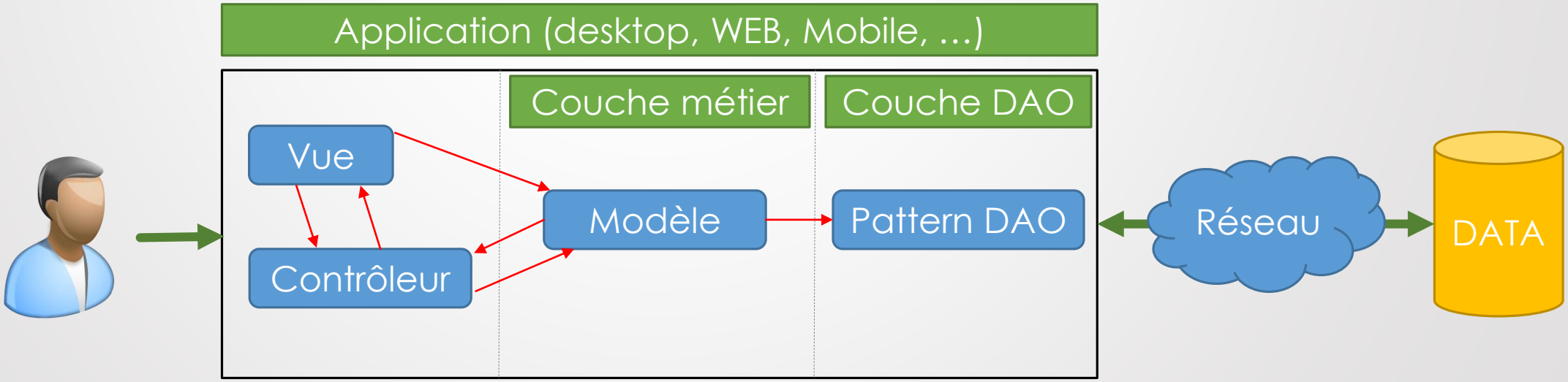
- Vue
- Contrôleur
- Modèle



I. Introduction

Une application est découpée en couche

- Vue
- Contrôleur
- Modèle
- Accès au données



I. Introduction

Pattern DAO

- Modèle pour concevoir des applications qui accèdent à des données
- Interface entre couche métier et données
- **Objectif**: séparer les couches métier et données
- Ne se limite pas aux bases de données
  - Fonctionne aussi avec des fichiers (XML, JSon, etc.)

Deux étapes

1. Préparer les classes
2. Mise en place des classes DAO

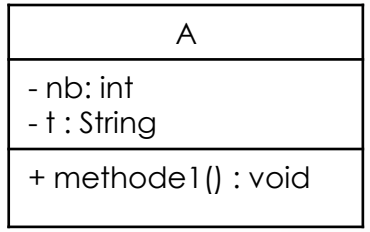
II. Implémenter les classes

Préparer les classes

- Mettre en place les classes pour quelles matchent au maximum avec les tables
- Respecter quelques règles sur l'implémentation des classes
- Rappel sur l'implémentation d'un diagramme de classe https://goo.gl/vxEXMZ

II. Implémenter les classes

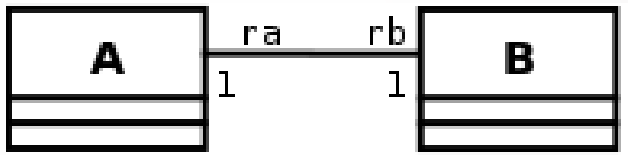
Association bidirectionnelle 1 vers 1




```
public class A {
	private int nb;
	private String t;
	public void methode1(){
		...
	}
}
```

II. Implémenter les classes

Association bidirectionnelle 1 vers 1

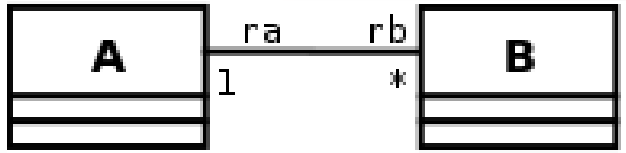



```java
public class A {
            private B rb;
}
```

```java
public class B {
            private A ra;
}
```

II. Implémenter les classes

Association bidirectionnelle 1 vers N



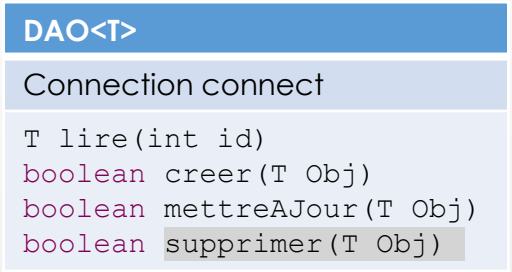

```java
public class A {
            private ArrayList<B> rb;
}
```

```java
public class B {
            private A ra;
}
```

III. Pattern DAO

Mise en place du Pattern DAO

- Pour chaque classe nécessitant un accès aux données
- Créer une classe permettant de réaliser les opérations CRUD
  - Créer (Create)
  - Lire (Read)
  - Modifier (Update)
  - Supprimer (Delete)
- Cadre donné par la classe abstraite DAO



III. Pattern DAO

Exemples avec deux classes

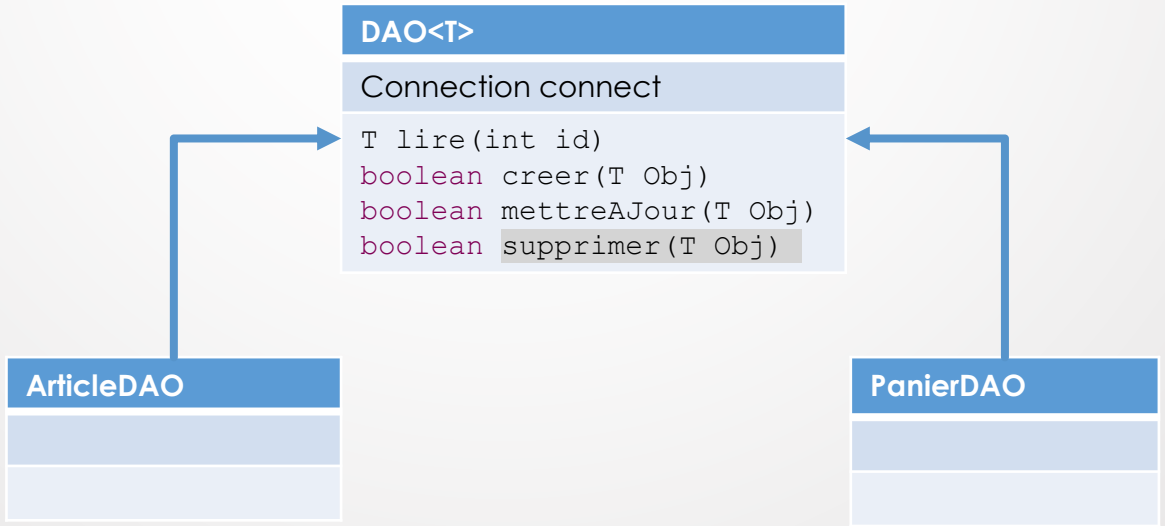
Article

```
int id;
String nom;
double prix
```

Panier

```
int id;
ArrayList<Article> liste;
```

- Pour chacune des classes, créer la classe DAO associée



III. Pattern DAO

Exemple de la classe ArticleDAO

```
public abstract class DAO<T>{
	protected Connection connect;
	public DAO(Connection c){
		connect = c;
	}
	//Lire un objet
	public abstract T lire(int id);
	//Creer un objet
	public abstract boolean creer(T Obj);
	//MAJ un objet
	public abstract boolean mettreAJour(T Obj);
	//Supprimer un objet
	public abstract boolean supprimer(T Obj);
}
```

```
public class ArticleDAO extends DAO<Article>{
	public ArticleDAO(Connection c) {
		super(c);
	}
	@Override
	public Article lire(int id) {
		Article a = null;
		try{
			ResultSet rslt = connect.createStatement().executeQuery(
			"SELECT id, nom, prix FROM article WHERE id="+id);
			//Si un article est trouvé dans la base
			if(rslt.next()){
				a = new Article(id, rslt.getString("nom"), rslt.getDouble("prix"));
			}
		}
		catch(Exception e){}
		return a;
	}
}
```

III. Pattern DAO

Accès aux données via la couche DAO

- Insérer un article dans la BD

```
Article a = new Article(3, "Tablette", 500);
ArticleDao bd = new ArticleDao(ma_connection);
bd.creer(a);
```

- Lire un article

```
ArticleDao bd = new ArticleDao(ma_connection);
Article a = bd.lire(3);
```

- Mettre à jour un article

```
ArticleDao bd = new ArticleDao(ma_connection);
Article a = bd.lire(3);
a.setPrix(480);
bd.mettreAJour(a);
```

III. Pattern DAO

Pour aller plus loin

- Ne se limite pas aux bases de données
- Peut être utilisé pour manipuler des données de d'autres sources
- Un même programme peut manipuler les données de plusieurs sources
  - Mettre en place une Factory

IV. Limites

Bilan

- Pattern DAO vous permet de lier les données aux objets
- Séparer la couche métier de la couche d'accès aux données
- Rend plus souple le changement de support

Limites ?

IV. Limites

Bilan

- Pattern DAO vous permet de lier les données aux objets
- Séparer la couche métier de la couche d'accès aux données
- Rend plus souple le changement de support

Limites ?

- Peut s'avérer complexe lorsque les objets sont très liés
- Implique un cout additionnel
- Plus grande complexité de mise en œuvre