Analyse, Conception Objet

J2EE Patterns

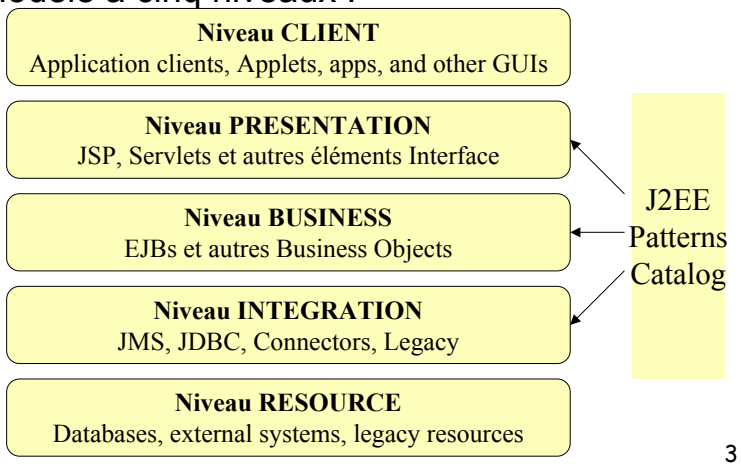
O. Boissier, SMA/G2I/ENS Mines Saint-Etienne, Olivier.Boissier@emse.fr, **Avril 2004**

Introduction

- Mis en place par SUN pour aider les développeurs à concevoir des applications en utilisant la technologie J2EE.
- Au nombre de 15, ils regroupent les meilleures pratiques de la technologie J2EE
- Distinction de 3 catégories distinctes correspondant aux niveaux présentation, métier, intégration d'une application
- http://java.sun.com/blueprints/patterns/index.html

Architecture multi-niveaux

Modèle à cinq niveaux :



Architecture multi-niveaux

- **Client** : ensemble des matériels ou systèmes clients accédant à l'application (Web Browser, Java, Java applet, WAP phone, …)
- **Présentation** : logique de présentation pour accéder au système (interception des entrées clients, gestion de session, contrôle les accès aux couches métiers) : servlets, jsp, ..
- **Métier (Business)** : services métiers requis par l'application, données métier et logique métier.
- **Intégration** : communication avec les ressources externes au système tels que BD, applications existantes, …
- **Ressource (Resource)** : données métier et ressources externes

Patterns de Niveau Présentation

- **Decorating Filter**
  facilitates pre and post-processing of a request.
- **Font Controller**
  provides a centralised controller for managing the handling of a request.
- **View Helper**
  encapsulates logic that is not related to presentation formatting into Helper components.
- **Composite View**
  creates an aggregate View from atomic sub-components.
- **Service To Worker**
  combines a Dispatcher component in coordination with the FrontController and View Helper Patterns.
- **Dispatcher View**
  combines a Dispatcher component in coordinating with the FrontController and View Helper Patterns, deffering many activities to View processing.

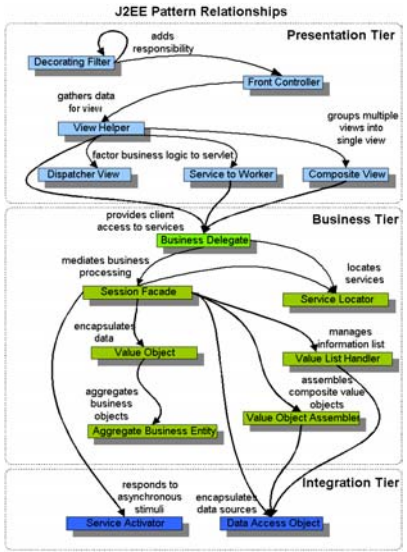
Pattern de niveau métier

- **Business Delegate**
  decouples presentation and service tiers, and provides a façade and proxy interface to the services.
- **Value Object**
  exchanges data between tiers.
- **Session Façade**
  hides business object complexity, centralises workflow handling.
- **Aggregate Entity**
  represents a best practice for designing coarse-grained entity beans.
- **Value Object Assembler**
  builds composite value object from multiple data sources.
- **Value List Handler**
  manages query execution, results caching & processing.
- **Service Locator**
  hides complexity of business service lookp and creation, locates business service factories.

Pattern de niveau Intégration

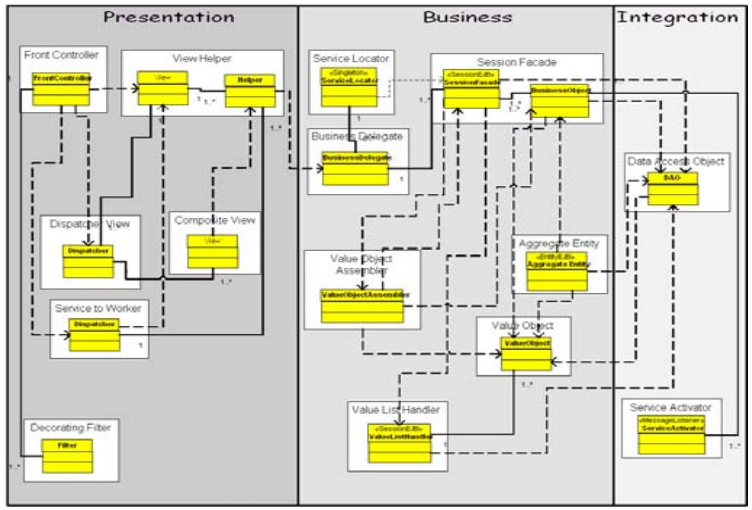
- **Data Access Object**
  abstracts data sources, provides transparent access to data.
- **Service Activator**
  facilitates asynchronous processing for EJB components.

Relations entre les patterns



Source: Sun Java Center J2EE Patterns

Vision Framework des Patterns



Source: Sun Java Center J2EE Patterns

Terminologie

- **BMP**
  - Bean managed persistence: a strategy for entity beans where the bean developer implements the persistence logic for entity beans.
  - used in: Business tier patterns
- **Business Object**
  - An object that implements business logic and/or business data. Business data and business logic are implemented in coarse-grained objects called Business Objects.
  - used in:Business tier patterns
- **CMP**
  - Container managed persistence: a strategy for entity beans where the container services transparently manage the persistence of entity beans.
  - used in: Business tier patterns
- **Composite**
  - A complex object that holds other objects.
  - used in: Composite View, Aggregate Entity

Terminologie

- **Controller**
  - Interacts with a client, controlling and managing the handling of each request.
  - used in: Presentation tier patterns, Business Tier
- **Data Access Object**
  - An object that encapsulates and abstracts access to data from a persistent store or an external system.
  - used in Business and Integration tier patterns
- **Delegate**
  - A stand-in for another component, an intermediate layer. A Delegate has qualities of a Proxy and Façade.
  - used in: Business Delegate and many other patterns
- **Dependent Object**
  - An object that does not exist by itself and whose lifecycle is managed by another object.
  - used in: Aggregate Entity pattern

Terminologie

- **Dispatcher**
  - Managing the choice of and dispatching to an appropriate View are some of the responsibilities of a Controller.
  - used in: Dispatcher View, Service To Worker
- **Enterprise Bean**
  - An instance of an Enterprise JavaBean component; can be a session or entity bean instance.
  - used in many places in this literature
- **Entity Bean**
  - Refers to an instance of an entity bean. May also refer collectively to the entity bean's home, remote object, bean implementation, or primary key objects.
  - used in many places in this literature
- **Façade**
  - A pattern to hide the complexities described in the Design Patterns book by Gamma et al.
  - used in: Session Façade pattern

Terminologie

- **Factory** (Abstract Factory or Factory Method)
  - Patterns described in the GoF book for creating objects or families of objects.
  - used in: Business tier patterns:Data Access Object, Value Object
- **Front**
  - Same as Controller.
  - used in: Presentation tier patterns
- **Iterator**
  - A pattern to provide collection facility described in the GoF book.
  - used in: Value List Handler
- **GoF**
  - Gang of Four--refers to the authors of the popular Design Patterns book (Gamma, Helm, Johnson, Vlissides)
  - used in many places in this literature

Terminologie

- **Helper**
  - Responsible for helping the Controller and/or View.
  - used in: Presentation tier patterns, Business delegate
- **Independent Object**
  - An object that can exist by itself and may manage the lifecycles of its dependent objects.
  - used in: Aggregate Entity pattern
- **Locator**
  - An object that aids in locating service and business objects.
  - used in: Service Locator pattern
- **Model**
  - A physical or logical representation of the system or its sub-system.
  - used in: Presentation and Business tier patterns
- **Persistent Store**
  - Represents persistent storage systems such as RDBMS, ODBMS, a file system, and so forth.
  - used in: Business and Integration tier patterns

Terminologie

- **Value Object**
  - An arbitrary Java object that is used to carry data from one object/tier to another. Usually does not contain any business methods. May be designed with public attributes or provided with get methods to obtain attribute values.
  - used in: Business tier patterns
- **View**
  - The view manages the graphics and text that make up the display. It interacts with Helpers to get data values with which to populate the display. Additionally, it may delegate activities, such as content retrieval, to its Helpers.
  - used in: Presentation tier patterns

References and Further Reading

- The Design Patterns Companion – Cooper http://www.patterndepot.com/put/8/JavaPatterns.htm
- The Sun Java Centre: http://developer.java.sun.com/developer/technicalArticles/J2EE/despat/
- Jim Coplien's History of Design Patterns http://www.bell-labs.com/user/cope/Patterns/ICSE96/node1.html