IFT6803:
Génie logiciel du commerce électronique

Chapitre 3: Conception orientée objet

Section 3: Conception détaillée

Julie Vachon,
Hiver 2003

Sommaire

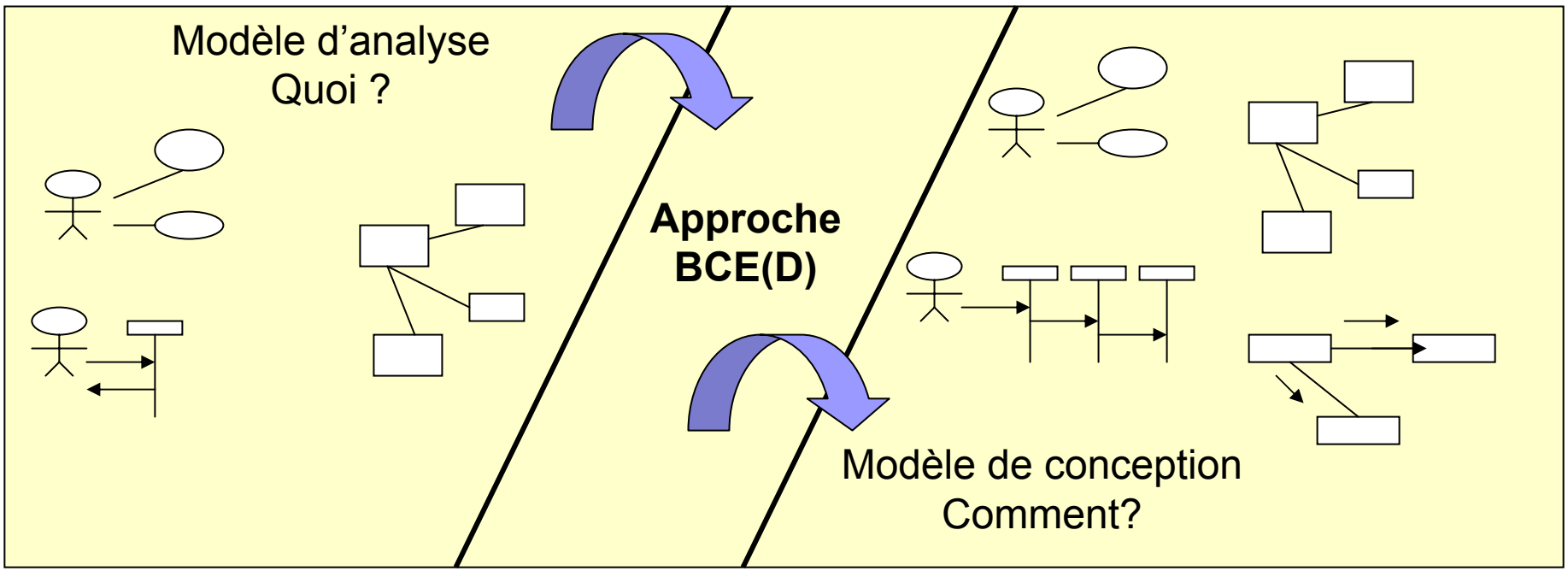
Chapitre 3, Section 3

« Conception détaillée »

3.3.1. Réalisation des collaborations

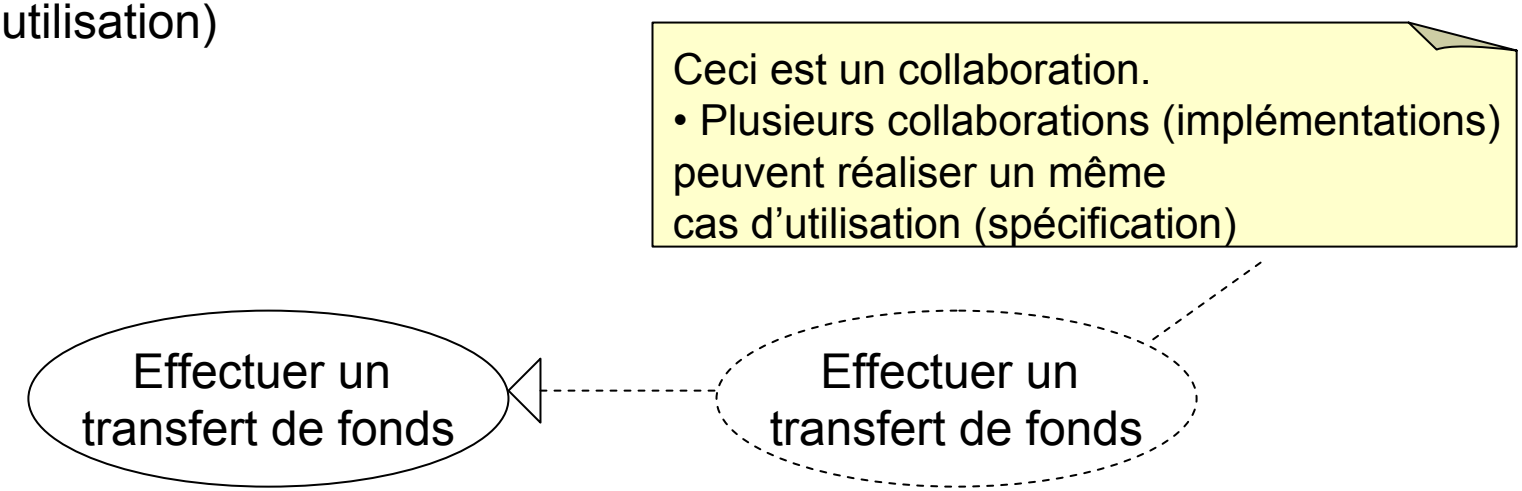
Objectifs de la conception détaillée

- Transformer le modèle d'analyse (spécification, haut niveau d'abstraction) en un modèle de conception détaillé (implémentation, bas niveau d'abstraction, concret, détails techniques) à partir du duquel le programmeur pourra directement implémenter le système.



3.3.1. Réalisation des cas d'utilisation

- Pour chaque cas d'utilisation, on cherche décrire comment il sera réalisé (implémenté).
- La conception détaillé a pour but d'associé une collaboration (implémentation) à chaque cas d'utilisation
- Collaboration =
  - Décrit la réalisation (l'implémentation) d'un cas d'utilisation (ou d'une opération).
  - Ensemble d'objets qui coopèrent pour réaliser une tâche (p. ex. cas d'utilisation)



Réalisation des cas d'utilisation

Collaboration

- Collaboration = Description d'un arrangement de liens et d'objets qui interagissent dans un contexte pour implémenter un comportement spécifié par un cas d'utilisation ou une opération.
  - **Partie statique**: instantiation des objets et des liens.
  - **Partie dynamique**: envoie de messages, calculs.
- Collaboration peut être décrite
  - par un **diagramme de collaboration** (statique & dynamique)
  - par un diagramme de classes (statique) & diagramme de séquence (dynamique)
  - Etc.

Réalisation des collaborations

Lien de réalisation

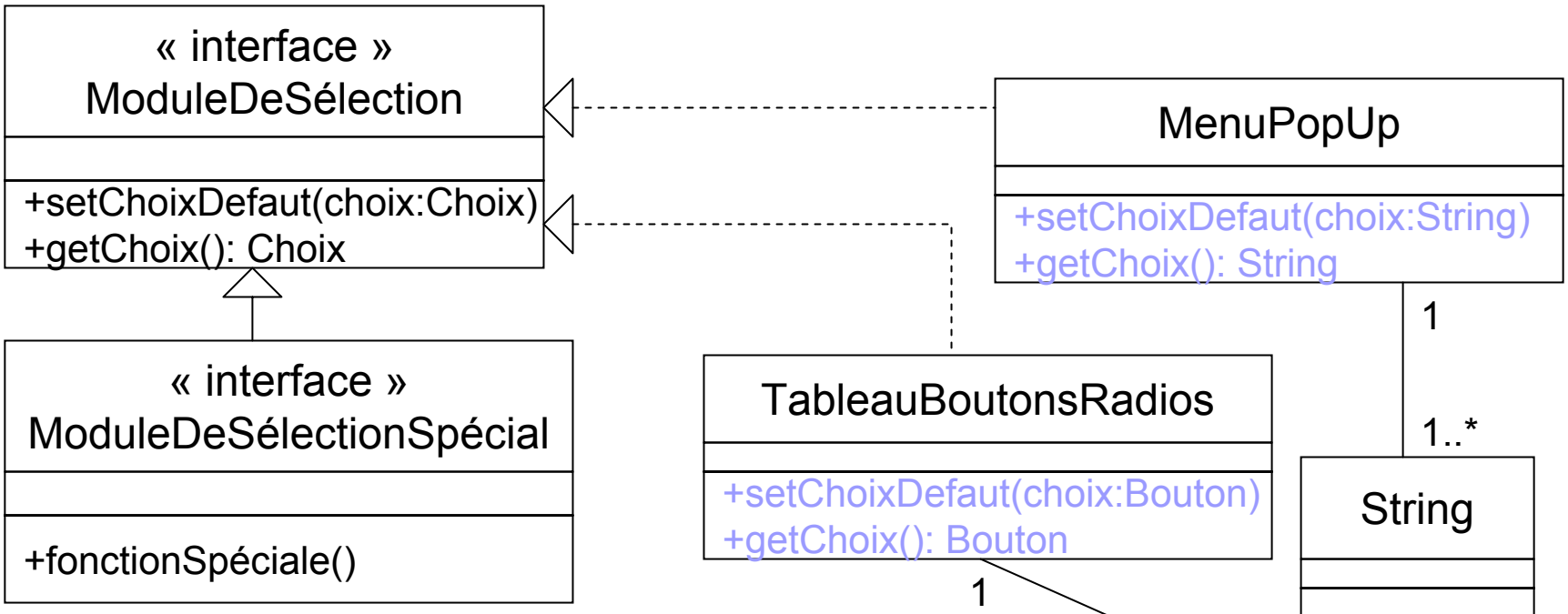
- Réalisation = Lien entre élément d'implémentation et élément de spécification.
  - Élément de spécification = Décrit le comportement et/ou la structure du système.
    - Ex. Interface, classe abstraite, cas d'utilisation
  - Élément d'implémentation = Explique en détail comment le comportement du système sera effectivement implémenté.
    - Ex. Classe, composant, collaboration.

Réalisation

~ héritage d'interface (et de comportement)

Réalisation des collaborations

Rappel: réalisation d'une interface par une classe



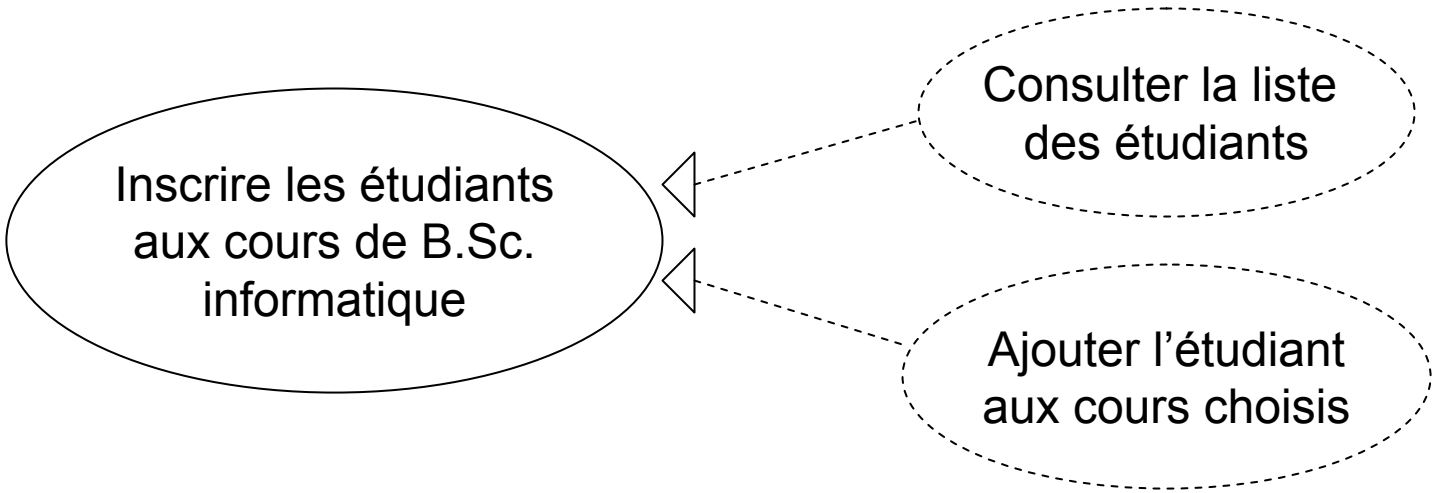
Interface = Collection d'opérations (pas de méthodes!) utilisées pour spécifier les services d'une classe ou d'un composant.

- Pas d'implémentation.
- Ne spécifie pas les attributs, associations, états.

setChoixDefaut et getChoix sont ici des opérations dites **surchargées** (overloading) et non redéfinies (overriding).

Réalisation des collaborations

Réalisation d'un cas d'utilisation par une (ou plusieurs) collaboration(s)



Réalisation des cas d'utilisation

Approche BCE(D)
pour construire le modèle de conception

1. Réviser la description détaillée de chaque cas d'utilisation. Ajouter les détails manquants.
2. Pour chaque cas d'utilisation, construire un diagramme de collaboration (ou de séquence):
    1. Identifier les objets « boundary », « entity », « control » et « databaseInterface » qui participent au scénario principal (et secondaires). Ajouter ces objets au diagramme de collaboration.
    2. Mettre à jour le diagramme de classes en ajoutant les nouvelles classes ainsi découvertes.
    3. Examiner plus en détails chaque objet « control » et tenter **d'attribuer des responsabilités** (méthodes) à chacun des objets collaborant dans le scénario
        1. Indiquer les messages sur le diagramme de collaboration.
        2. S'assurer que les interactions respectent les règles.
    4. Compléter le diagramme de classes en ajoutant les opérations correspondantes.
3. Réviser le modèle de conception et le ré-évaluer.
4. Regrouper les classes en paquetages.

Réalisation des cas d'utilisation

Approche BCE(D)

Attribution de responsabilités aux objets

- Distinguer les contrôleurs qui servent à gérer la présentation de ceux qui gèrent les fonctions applicatives du cas d'utilisation.
  - Identifier de façon plus précise les contrôleurs qui seront réalisés par des servlets, des pages JSP, etc.
- Préciser les responsabilités des objets « boundary » et leur types (formulaire, page client, etc.)
- Préciser la sémantique des objets « entity » et leur interactions avec les objets « databaseInterface ».
- S'inspirer des patterns de conception pour définir de bonnes structures et interactions.
- Attribuer les responsabilités de façon à ce que les classes soient faiblement couplées, fortement cohésives, suffisantes et complètes.

Réalisation des cas d'utilisation

Approche BCE(D)

Les interactions entre objets BCE doivent généralement respecter les règles suivantes:

- Les acteurs ne peuvent envoyer des messages qu'à des objets « boundary ».
- Les objets « boundary » ne peuvent envoyer des messages qu'à des objets « control » et aux acteurs.
- Les objets « entity » ne peuvent envoyer des messages qu'aux objets « control »
- Les objets « control » peuvent envoyer des messages aux objets « boundary », « control » et « entity », mais non aux acteurs.

P.S. Les objets « databaseInterface » peuvent être considérés comme un type particulier d'objets « control » ou « boundary »

Réalisation des cas d'utilisation

Approche BCE(D)

Révision du modèle de conception

- S'assurer que chaque message apparaissant dans le diagramme de collaboration (ou de séquence) correspond à un énoncé dans le scénario du cas d'utilisation.
- Vérifier la continuité du flot de contrôle, les bandes d'activation, la numérotation des messages.
- Ré-évaluer le couplage, la cohésion, la suffisance, la complétude et l'emploi de primitives.
  - Consolidation: fusionner les classes qui semblent redondantes. Ex. Deux classes « control » présentant des comportements similaires. Classes « entity » présentant les mêmes attributs.
  - Partitionnement: Diviser les classes comportant trop d'opérations en classes plus petites (meilleure maintenabilité)
- Le modèle de conception devrait refléter la façon dont les éléments de la technologie d'implémentation (EJB, DCOM, JSP, etc.) seront utilisés.

Réalisation des cas d'utilisation

Approche BCE(D)

Regrouper les classes en paquetages

- Paquetages
  - Permet de mieux gérer la complexité d'une application en regroupant les classes conceptuellement similaires ensembles.
    - N.b. Dans un composant, les éléments sont groupés non pas pour leur similitude conceptuelle mais parce qu'ils collaborent à la réalisation d'une tâche unique commune.
  - Des équipes différentes peuvent développer des paquetages distincts de façon relativement indépendante.
  - Identifier les relations de dépendance entre paquetages.

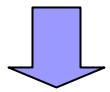
    Le paquetage X dépend du paquetage Y si, par exemple, une classe A de X utilise une classe B de Y.

3.3.2 Les servlets

Introduction

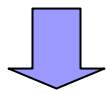
Les responsabilités des objets « control » sont souvent de deux natures:

- Interactions avec objets « boundary » (logique de présentation)
- Interactions avec objets « entity » (fonctions de l'application)



Si on combine tous dans une même classe « control »

- Risque de faible cohésion
- Difficulté de maintenance



Ainsi on propose de partitionner une classe « control » en deux (ou plusieurs) classes:

- Une responsable des interactions externes : SERVLET
- Les autres responsables de la coordination et de la logique interne applicative.

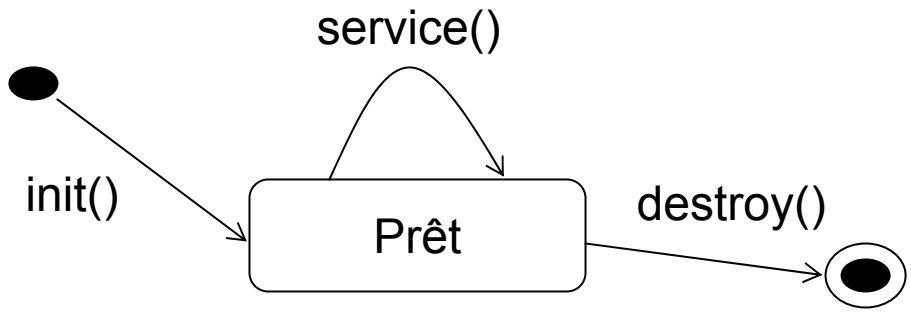
Les servlets

Servlet

- Conçu spécifiquement pour traiter les requêtes HTTP des clients Web.
- Classe Java spécialisées.
- Logé dans un container d'un serveur Web qui en gère le cycle de vie.
- Composant approprié pour réaliser un ensemble d'exigences relativement simple. Intermédiaire entre client et serveur web.
  - Réunir et valider les entrées d'un utilisateur (provenant d'un formulaire par exemple)
  - Coordonner les sorties (sans génération de page web dynamique…)
  - Logique applicative minimale.
- Pas recommandé pour réaliser des interactions Web nombreuses et complexes (trop bas niveau). Autres solutions: JSP, EJB, J2EE, etc.

Les servlets

Cycle de vie d'un servlet



Opérations de base d'un servlet

- **init**: Initialise le servlet
- **service**: exécute la requête du client
  - Recoit la requête du client
  - Lit les données de la requête
  - Rédige l'entête de la réponse
  - Propose un objets writer ou output stream pour la réponse
  - Ecrit les données de la réponse.
- **destroy**: détruit le servlet.

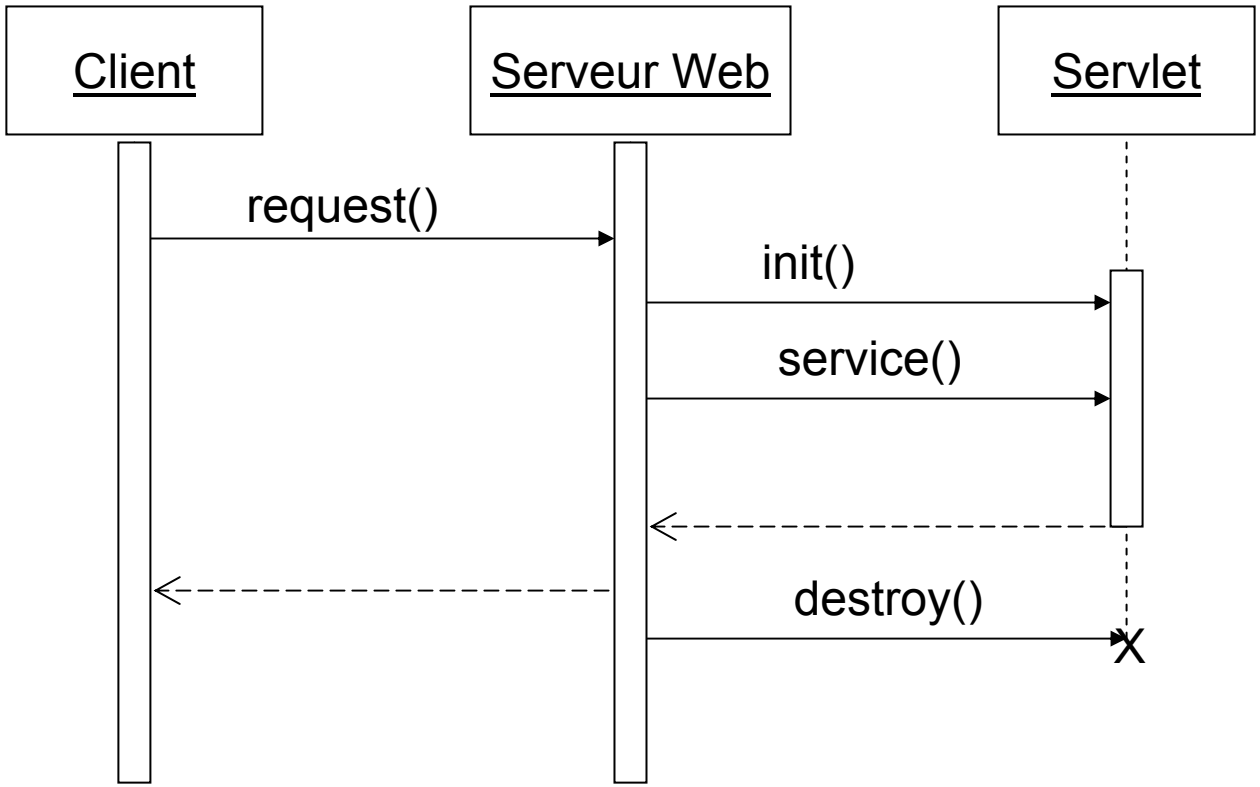
Les servlets

Types de servlets

- GenericServlet
  - http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api/javax/servlet/GenericServlet.html
- HttpServlet
  - http://java.sun.com/j2ee/sdk_1.2.1/techdocs/api/javax/servlet/http/HttpServlet.html
  - Sous-classe de GenericServlet.
  - Méthode service raffinée pour tenir compte de la spécificité des requêtes HTML.

Les servlets

Utilisation d'un servlet



Les servlets

Requête traitée par un servlet

GenericServlet
+ init() + destroy() **+ service(req:ServletRequest, res:ServletResponse)**

- Lorsque le container Web reçoit une requête, celle-ci est encapsulée dans un objet de type **ServletRequest** et passée en paramètre à l'opération **service** du servlet.

ServletRequest
+ getCharacterEncoding(): String + getParameterNames(): Enum + getParameter(p:string): String + getParameterValues(p:String): StringArray …

Les servlets

Réponse produite par un servlet

GenericServlet
+ init() + destroy() **+ service(req:ServletRequest, res:ServletResponse)**

- La réponse à une requête peut être écrite
  - en binaire : référence sur un objet ServletOutputStream nécessaire
  - en caractères : référence sur un objet PrintWriter nécessaire.

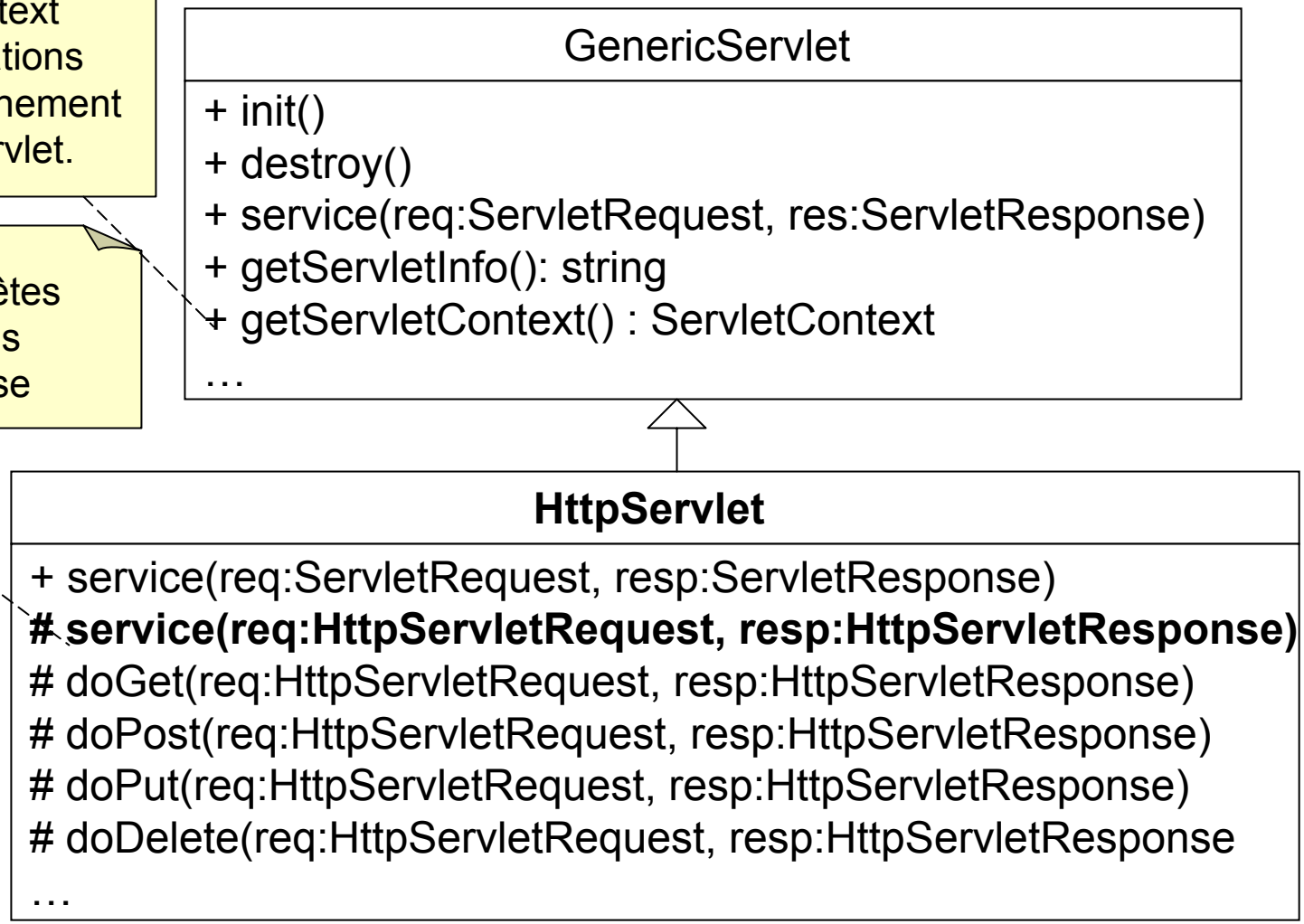
ServletResponse
+ getOutputStream(): ServletOutputStream + getWriter(): PrintWriter + setBuffersize(i:Integer): String + flushBuffer() …

Les servlets

Types de servlets

L'objet ServletContext fournit des informations décrivant l'environnement où s'exécute le servlet.

Redirige les requêtes vers les opérations doXXX de la classe



Les servlets

Types de requêtes HTTP

- **GET**: Un appel pour obtenir une information du serveur.
- **POST**: Un appel pour permettre au client d'envoyer des données au serveur.
- **PUT**: Similaire à POST, mais permet au client de déposer un fichier sur le serveur.
- **DELETE**: Similaire à PUT mais permet au client de retirer une page web du serveur.

Les servlets

Collaboration entre servlets

Si on souhaite garder les servlets simples, on préfèrera les faire **collaborer** pour réaliser une tâche complexe.

Comment préparer les servlets pour qu'ils puissent collaborer ?

RequestDispatcher:

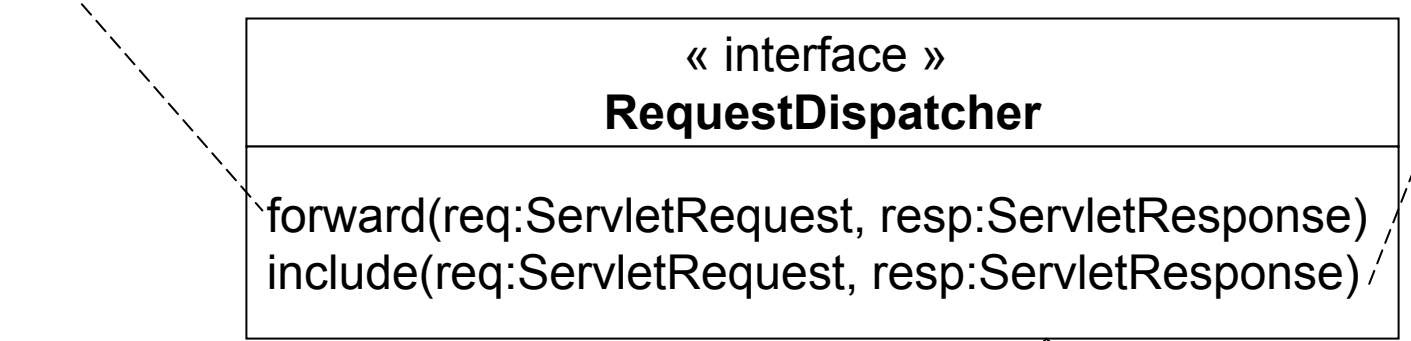
- Interface crée par un container web pour un servlet.
- Enveloppe un servlet pour lui permettre de collaborer avec d'autres ressources (servlets)

Les servlets

Collaboration entre servlets

Forward:
Permet à un servlet de faire suivre la requête à un autre composant pour qu'il en complète le traitement. Travail en pipeline.

Include:
Permet à un servlet d'inclure la réponse d'un autre composant dans sa propre réponse.



Les servlets

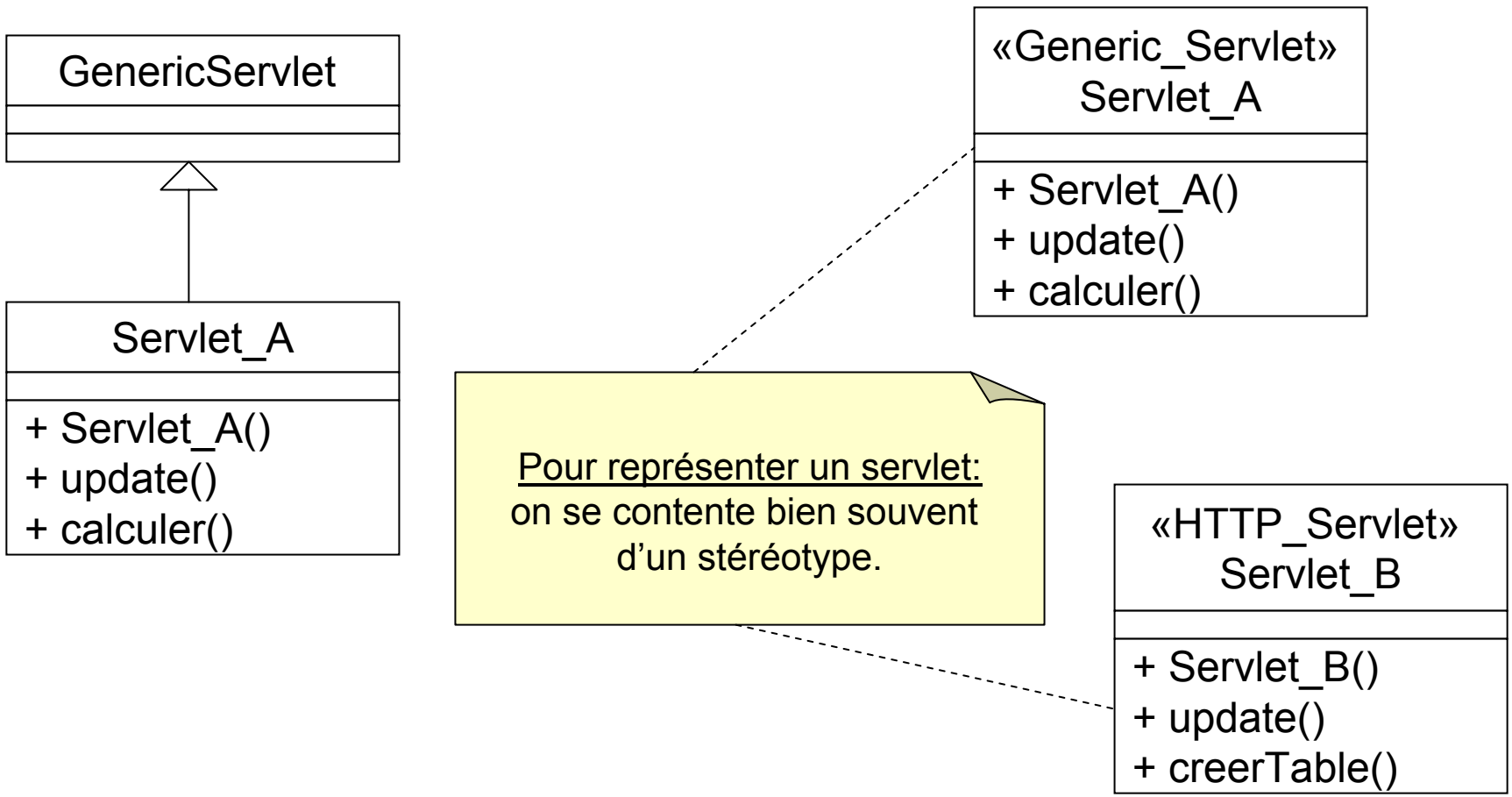
Les servlets dans UML

On peut spécifier l'utilisation de servlet dans le design détaillé UML d'une application.

- Il n'est certes pas nécessaire de représenter les classes et interfaces de la librairie Java utilisées pour spécifier l'utilisation d'un servlet en UML !!! On ne représente que les classes qu'on développe soi-même pour l'application.
  - Ex. Inutile de représenter la classe RequestDispatcher.
- On utilise des stéréotype pour spécifier la nature particulière des classes représentant des servlets et des associations entre elles.
  - Stéréotypes utilisés pour les classes
    - « Generic_Servlet », « Http_Servlet »
  - Stéréotypes utilisés pour les associations
    - « include », « forward »

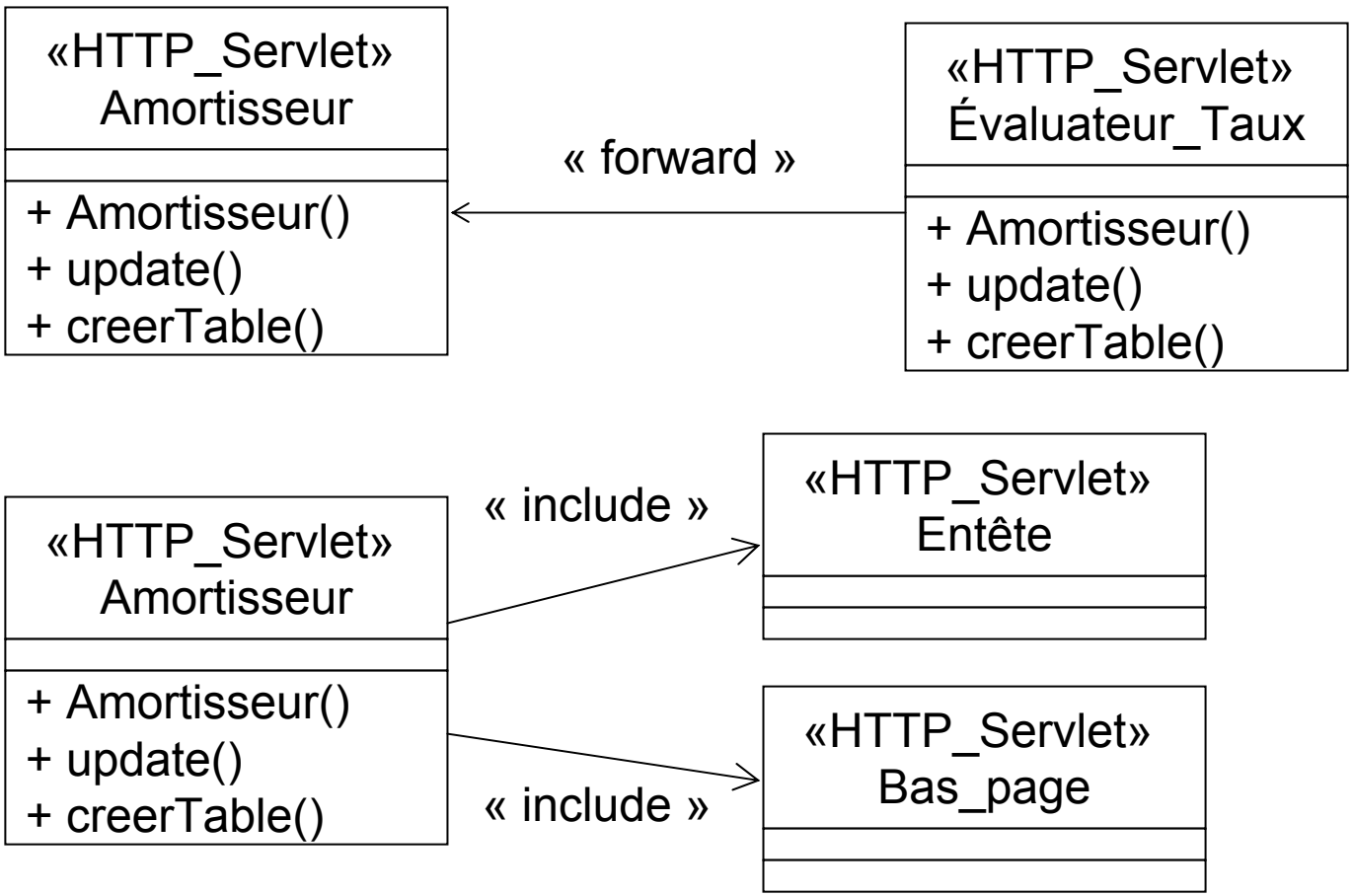
Les servlets

Les servlets dans UML



Les servlets

Les servlets dans UML



Les servlets

Quand utiliser un servlet dans une application ?

Pendant le raffinement de la phase de conception détaillée, on pourra proposer l'utilisation d'un ou plusieurs servlets pour réaliser toutes ou une partie des responsabilités (fonctions) assumées par un objet « control » du système.

- On privilégiera l'utilisation de servlets pour la coordination des interactions avec l'extérieur du système.
- La façon de séparer les responsabilités d'un objet « control » entre deux ou plusieurs servlets dépend du contexte.
- On essaie de minimiser les responsabilités associées à chaque servlet.

3.3.3 Java Server Page

Qu'est-ce qu'une page JSP ?

- Document structuré dans lequel se trouve imbriqué du code Java:
  - **Partie statique**
    - Gère la portion statique de la présentation
    - HTML, XML, etc.
  - **Partie dynamique**
    - Délimitée par des tags.
    - Compilée (en servlet) pour exécution par la partie serveur.
- Remarque: Éviter d'utiliser une page JSP pour représenter le contenu d'une page statique. La compilation de la page entraîne alors des coûts inutiles.

Java Server Page

JSP et servlets

- Les JSP ont les mêmes «capacités » que les servlets.
- Les JSP sont des composants centrés sur la présentation.
- L'emploi de JSP permet de séparer la présentation de la logique applicative.
- Les JSP facilitent l'organisation des aspects physiques d'une application Web.
- Les JSP sont compilées automatiquement lors du déploiement.
- On recommande une utilisation mixte des deux technologies
  - JSP
    - À utiliser pour la **présentation** dynamique des documents. Ex. Mise à jour dynamique du contenu d'une page.
  - Servlet
    - À utiliser pour la **logique de l'application**.

Java Server Page

Chargement d'une page web (JSP)

1. Lorsque une page web est demandée, le serveur web charge la fichier contenant la page et exécute les fonctions définies dans sa partie *serveur* et produit éventuellement un flux de sortie (HTML, XML, etc.)
2. Le flux de sortie est accepté par le navigateur du client qui l'affiche.
3. Si la partie *cliente* contient un script (java) et que le navigateur client est en mesure de l'exécuter, les fonctions définies dans la partie *cliente* sont exécutées (handlers d'événements) .

Java Server Page

Une page JSP est composée de

- Données de template
  - Décrivent les aspects statiques.
  - Partie XML, HTML du document.
- Éléments JSP
  - Décrivent les aspects dynamiques.
  - Partie du document compilée en servlet
  - Délimités par des tags
  - Types d'éléments
    - Les directives
    - Les actions
    - Les scripts

Java Server Page

Éléments JSP

1. **Éléments de directive**
   - Information globale pour la compilation.
   - Ex. **<%** include file=```Header.jsp `` **%>**
2. **Éléments d'action**
   - Action exécutée lors du traitement de la JSP.
   - L'action consiste à exécuter le tag-handler associé au tag rencontré. *Le tag-handler est (pré-)défini dans une librairie de tags.*
   - Actions standards (tags préfixées par jsp)
     - Include: inclure les réponses envoyées par les autres JSP
     - Forward: faire suivre les requêtes à un autre JSP.
     - Query et update: consulter et mettre à jour les propriétés des JavaBeans résidant sur le serveur.
   - Ex: **<jsp:forward** page =``/errorPage`` **/>**

Java Server Page

Éléments JSP

3. Éléments de script

- Déclarations (de variables, de méthodes)
  - Ex: **<%!** private static MyLoginCount=0; **%>**
- Expressions
  - Évaluées lors du traitement de la JSP.
  - Ex: Compteur de login: **<%=** results **%>**

Java Server Page

Éléments JSP

- Scriptlets
  - Mini – script composé de déclarations (variables et méthodes), d'expressions et d'énoncés.
  - Exécuté lors du traitement de la JSP.
  - Résultat placé dans l'objet réponse.
  - Ex.

```
<% int guessNum = request.getParameter(« GUESS »);
If (guessNum == Winning Num) {%/>
Congrats, you win !!!
<%}
Else
{ %/>
Sorry, try again.
<%} %/>
```

Java Server Page

Objets accessibles implicitement par toute JSP

Objets créés par le container

- request
- response
- pageContext
- session
- application
- out
- config
- page
- exception

Java Server Page

Approches de conception avec JSP

- **Approche 1:** Utilisation de JSP à la fois pour la présentation et pour la logique applicative.
  - Solution simple.
  - Implémentation risque d'être difficile à gérer.
- **Approche 2 :** Inspirée du paradigme MVC. Utilise un ou plusieurs **servlets pour la logique applicative** et **JSP pour la présentation**.
  - Choisir le bon nombre de servlets pour réaliser les tâches.
    - Cas extrêmes: un par cas d'utilisation // un seul pour toute l'application.
  - Utilisation de JavaBeans comme unité de communication entre les servlets et composants JSP.
  - Séparation claire entre présentation et logique applicative.
  - Solution plus complexe.
  - Implémentation plus facile à gérer

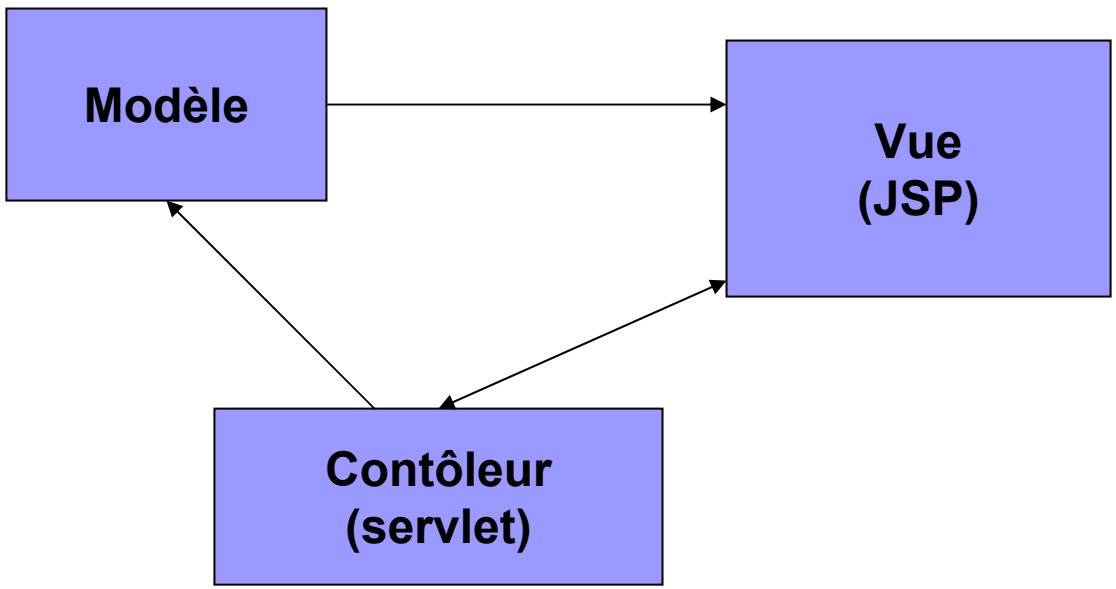
Java Server Page

Approche 2 – MVC

Principe: Séparation des deux mécanismes

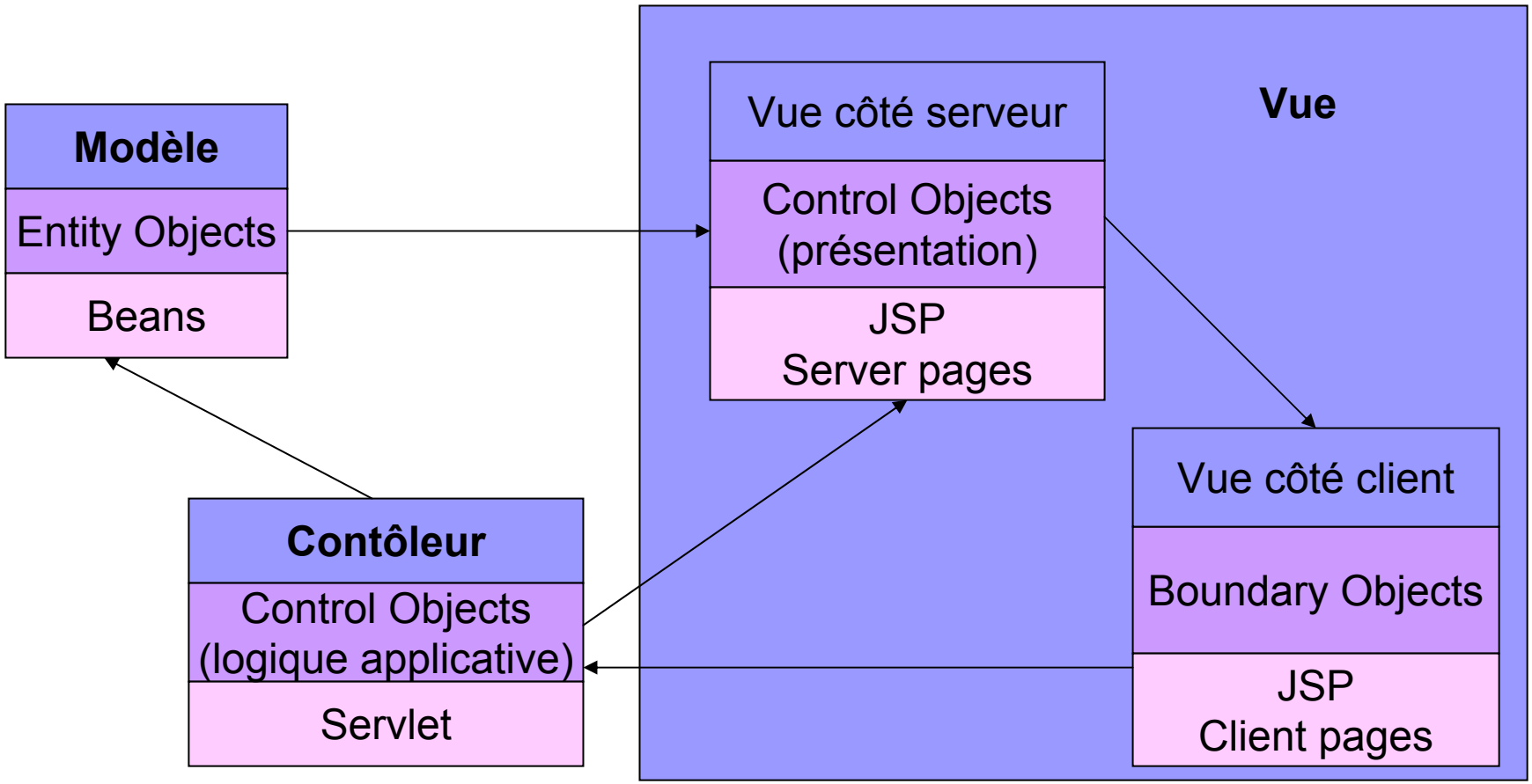
(1) accepter et traiter les entrées (servlet)

(2) construire les sorties (page jsp)



Java Server Page

Approche 2 - MVC



Java Server Page

Modélisation des JSP en UML

Une page JSP

- Représentée logiquement par une classe.
- Mais pour mettre en évidence la séparation des responsabilités entre les parties client et serveur de la page, une page JSP est modélisée par deux éléments conceptuels:
  - Une page serveur
  - Une page client

On réifie la partie la partie serveur de la page. Ainsi, une page Web JSP se trouve conceptuellement composée de deux structures.

Java Server Page

Modélisation des JSP en UML

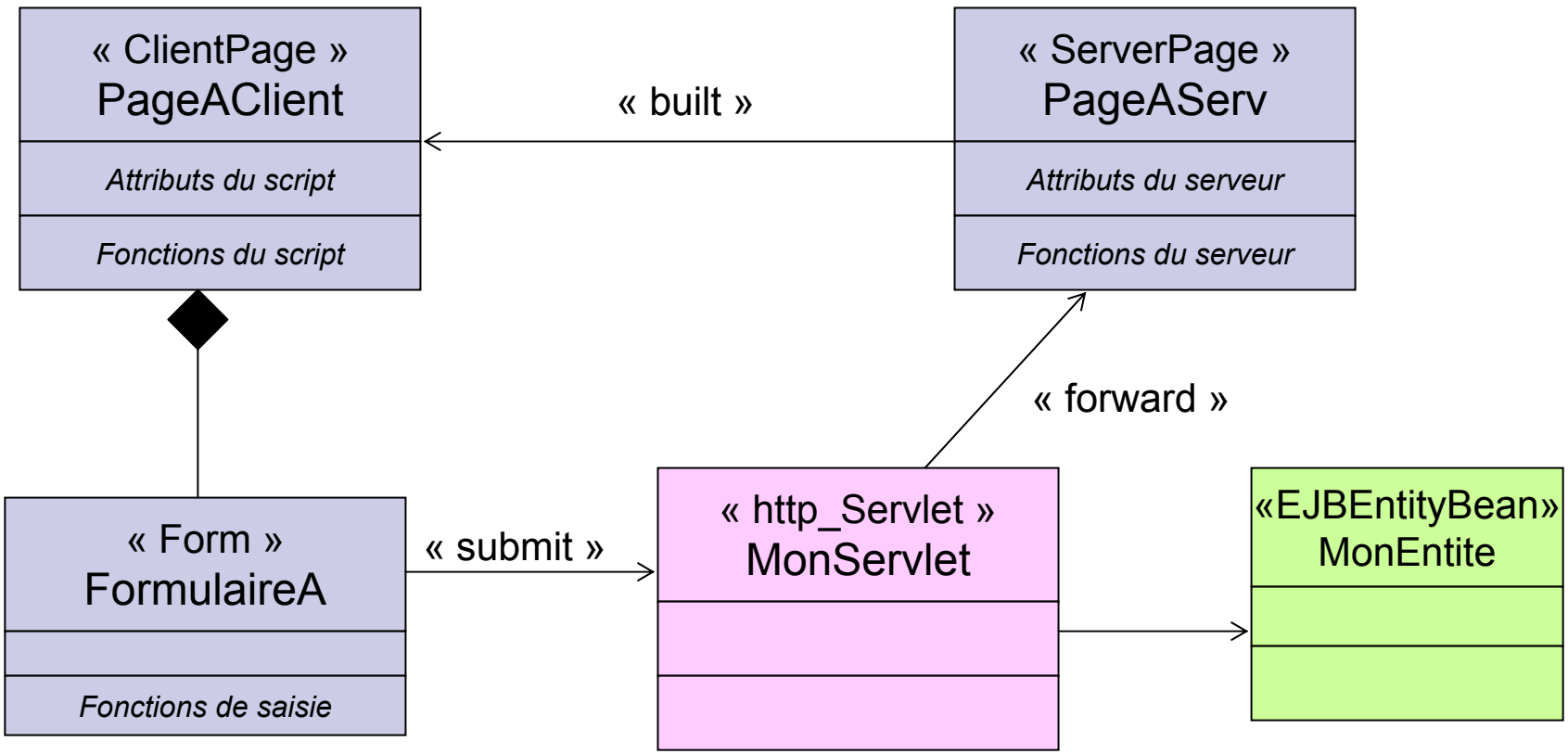
- **« ServerPage »** (logique applicative, accès aux bases de données, aux autres composants, aux ressources externes, etc.)
  - Partie comportement de la JSP. Exécutée sur le tiers serveur.
  - Attributs, opérations
  - Associations possibles avec: «ServerPage», «ClientPage», «servlet », « JavaBeans », etc.

- **« ClientPage »** (HTML + logique de présentation)
  - Partie présentation visible de la JSP. Exécutée sur le tiers client.
  - Attributs, opérations.
  - Associations possibles avec: «ClientPage», «Java Applet», «Form» «JavaBean»

Java Server Page

Modélisation des JSP en UML

Approche 2 - MVC



Java Server Page

Modélisation des JSP en UML

Stéréotypes pour définir certains types particulier d'associations…

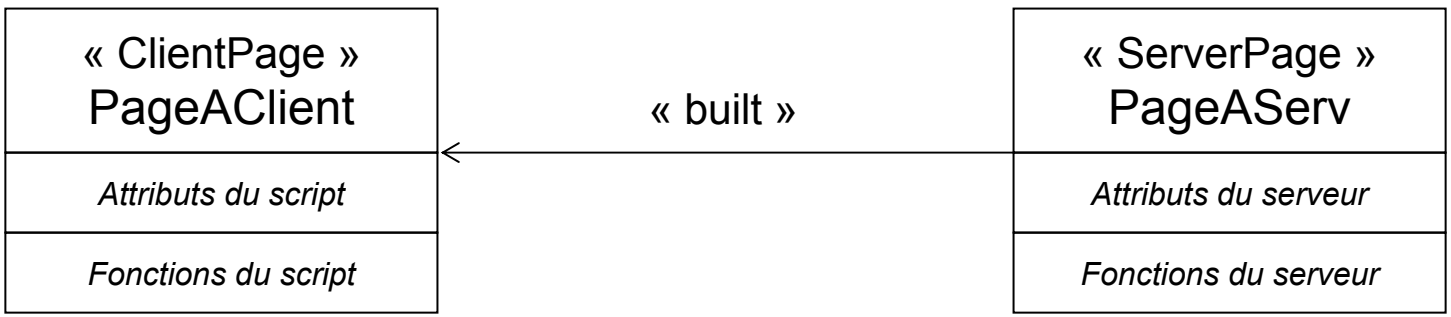
- **« build »**
- **« link »**
- **« submit »**
- **« include »**
- **« forward »**
- **Etc.**

Java Server Page

Modélisation des JSP en UML

Relations entre «ClientPage» et «ServerPage»

- Association unidirectionnelle avec stéréotype **«build»** : Un flux HTML ou XML est envoyé au navigateur client qui a émis la requête.



Java Server Page

Modélisation des JSP en UML

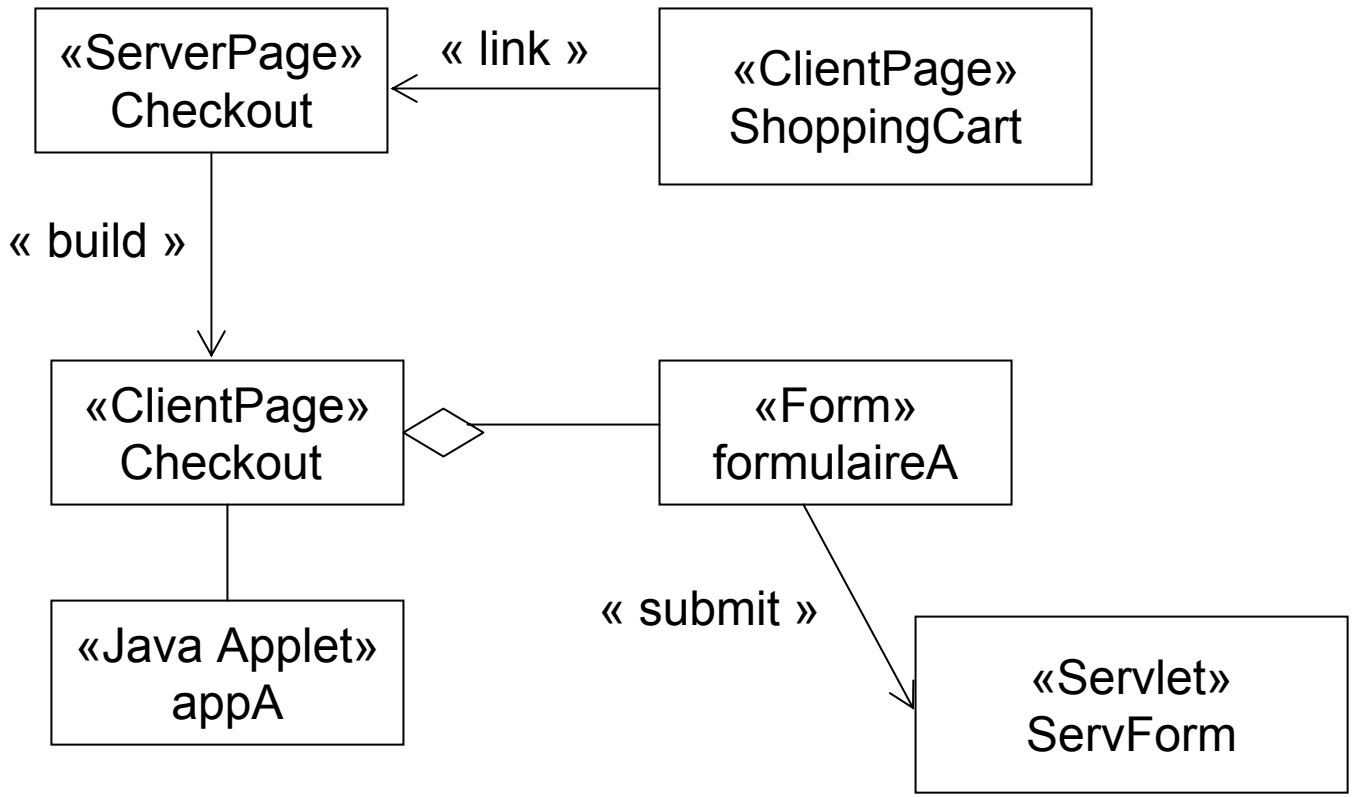
Relations d'une «ClientPage» avec…

- Autre page JSP.
  - Relation unidirectionnelle avec stéréotype **«link»** : Hyperlien d'une «clientPage» vers une autre page.
- Applet Java
  - Agrégation ou association simple d'une page client avec classe représentant une applet java.
- Formulaire
  - Agrégation du formulaire à la page client.
  - Association stéréotypée **«submit»** du formulaire à la page serveur responsable de son traitement.

Java Server Page

Modélisation des JSP en UML

Relations d'une « ClientPage »



Java Server Page

Modélisation des JSP en UML

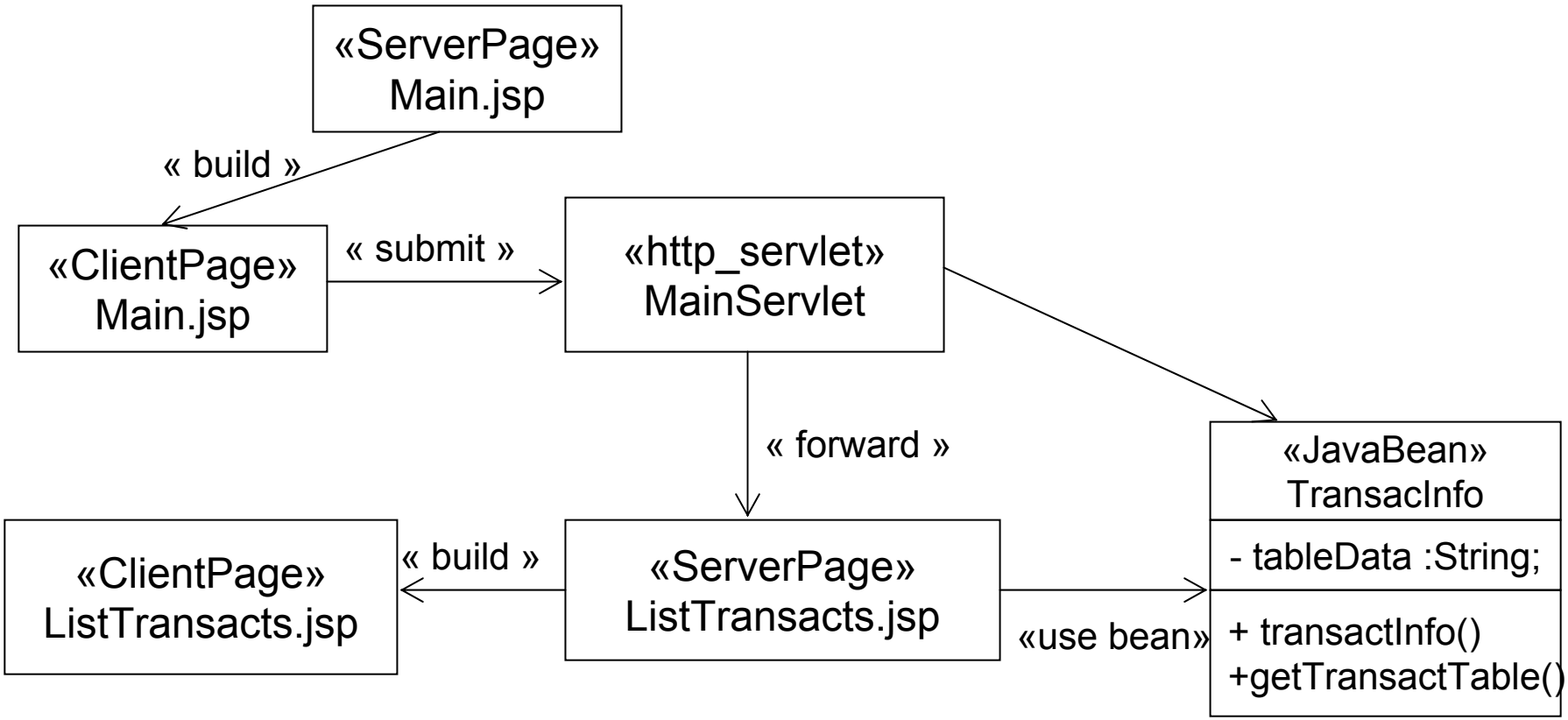
Relations d'une «ServerPage» avec…

- Autre « ServerPage » ou avec un servlet.
  - Association unidirectionnelle avec stéréotype **«include»** ou **«forward».**
- Java Beans ou EJB.
  - Association unidirectionnelle stéréotypée **«use bean».**
- Autres classes, librairies, etc.

Java Server Page

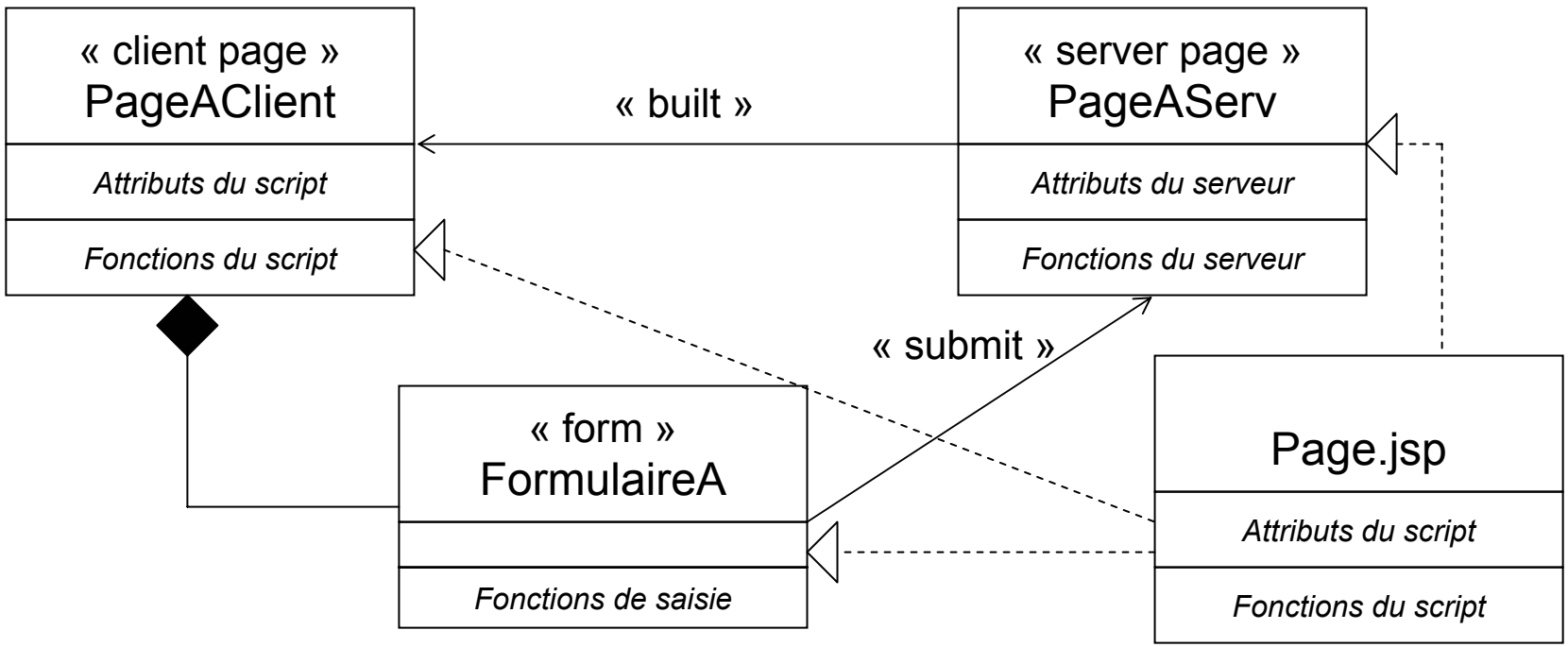
Modélisation des JSP en UML

Relations d'une « ServerPage »



Java Server Page

Réalisation d'une page web (JSP),



3.3.4.1 EJB – Session Beans

Modélisation des Session Beans en UML

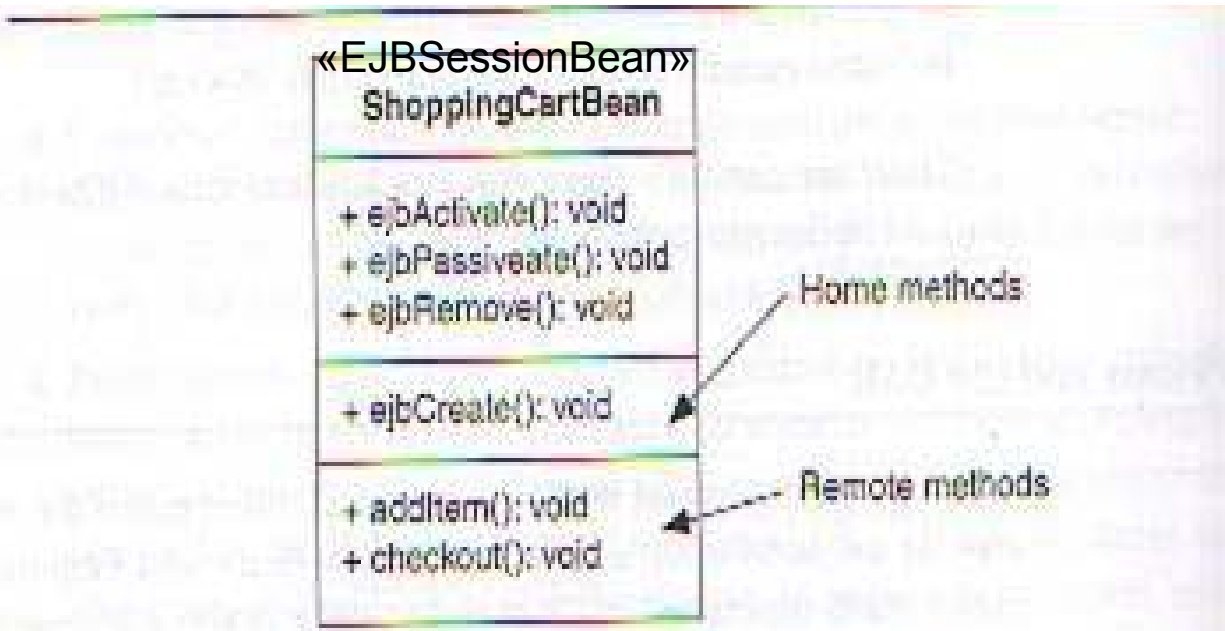


Figure 12-1 Example of a class-based EJB representation

Session Beans

Modélisation des Session Beans en UML

Vue Client

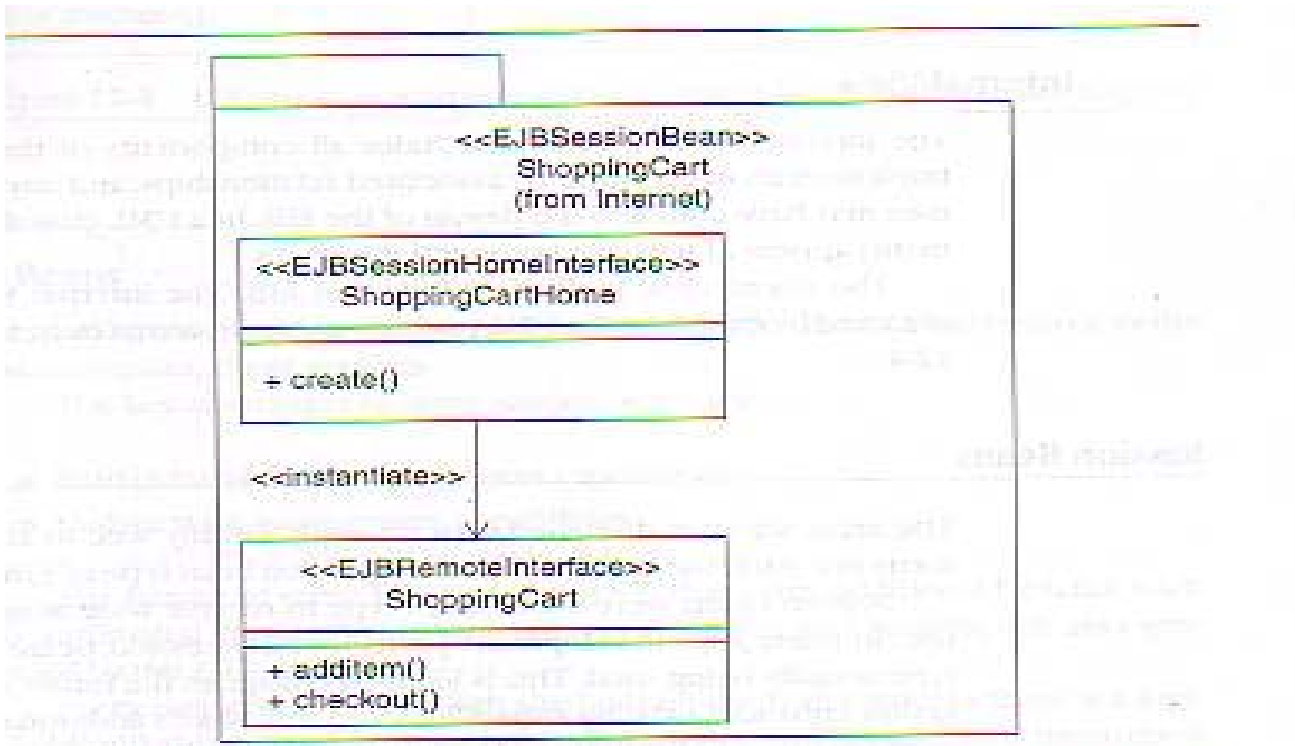


Figure 12-2 Full client view representation of a session bean in a UML class diagram

Session Beans

Modélisation des Session Beans en UML

Vue Client

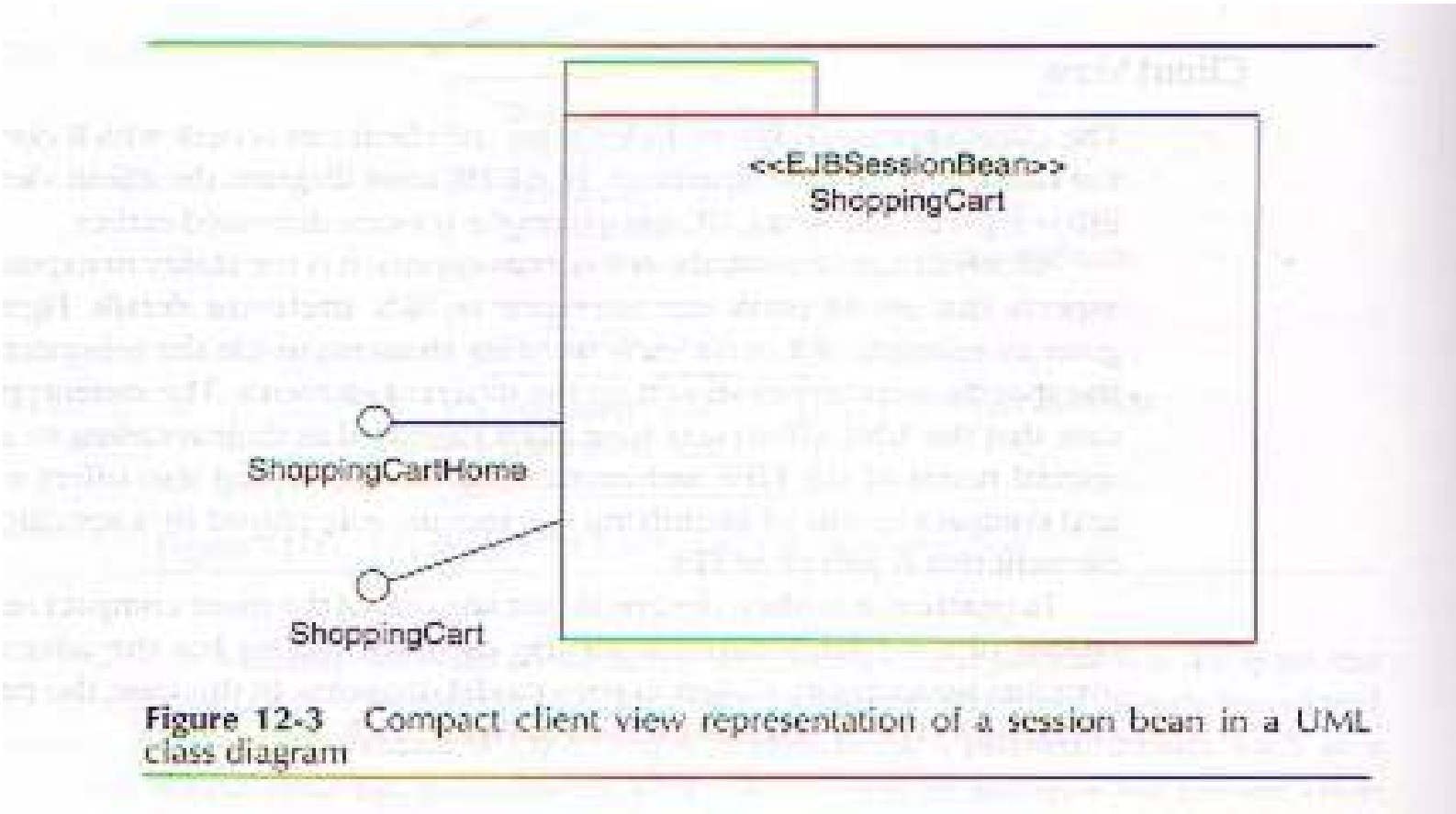


Figure 12-3 Compact client view representation of a session bean in a UML class diagram

Session Beans

Modélisation des Session Beans en UML

Vue interne

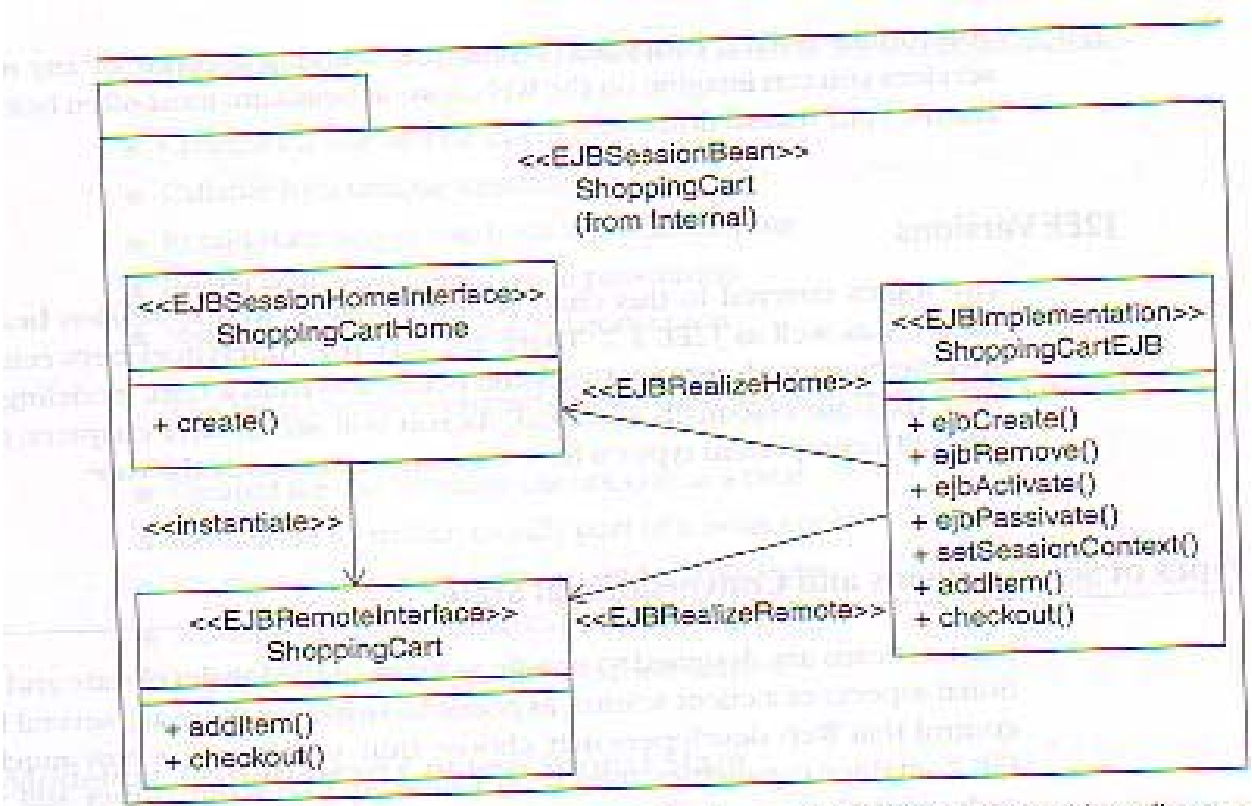


Figure 12-4 Internal view representation of a session bean in a UML class diagram

Session Beans

Modélisation des Session Beans en UML

Diagramme d'état du session bean «ShoppingCart»

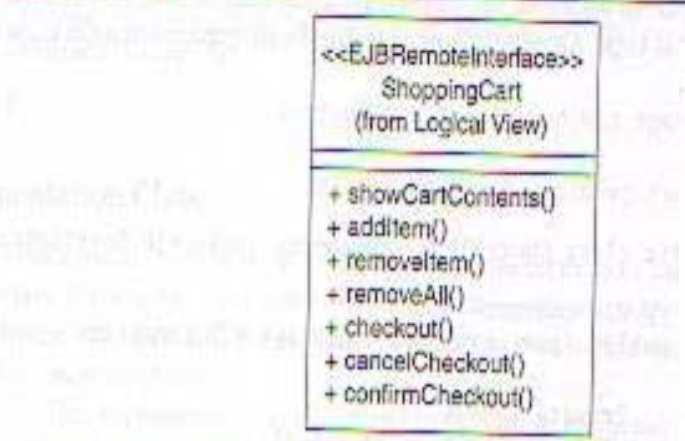


Figure 12-8 ShoppingCart session bean Remote interface

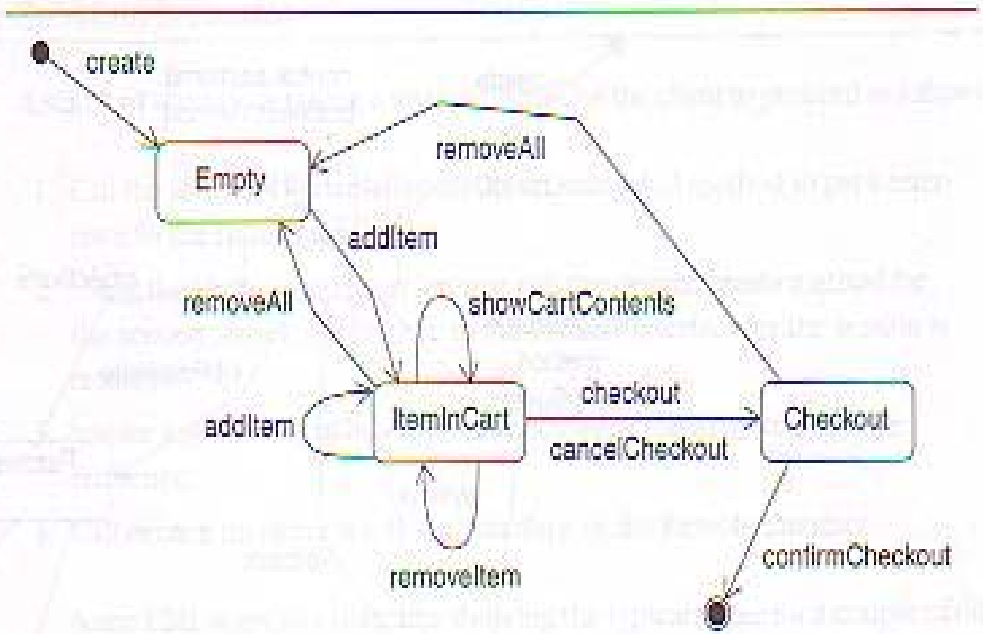
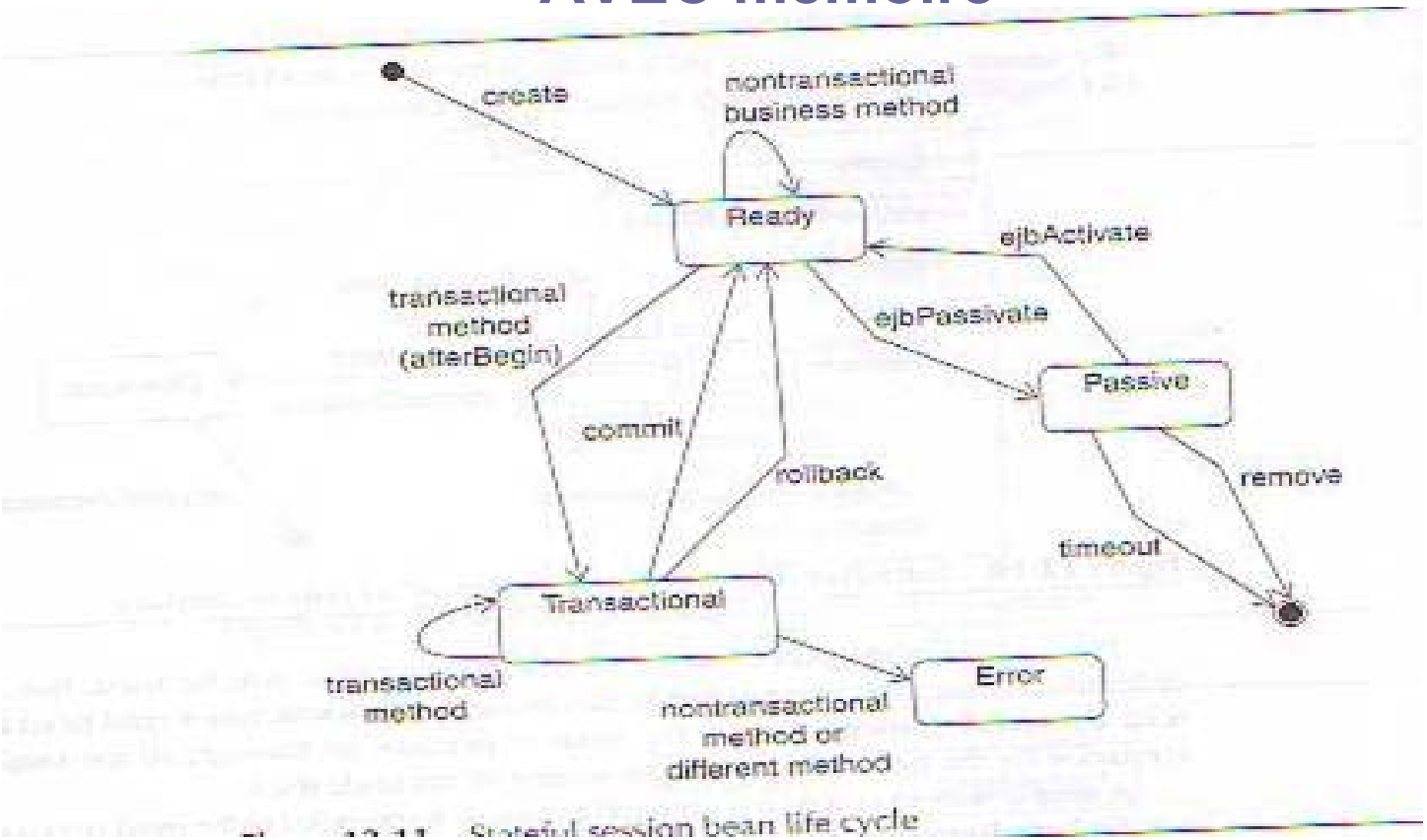


Figure 12-10 Statechart diagram for the ShoppingCart remote interface

Session Beans

Modélisation des Session Beans en UML

Diagramme d'état (général) d'un session bean AVEC mémoire



Stateful session bean life cycle

Session Beans

Modélisation des Session Beans en UML

Diagramme d'état (général) d'un session bean SANS mémoire

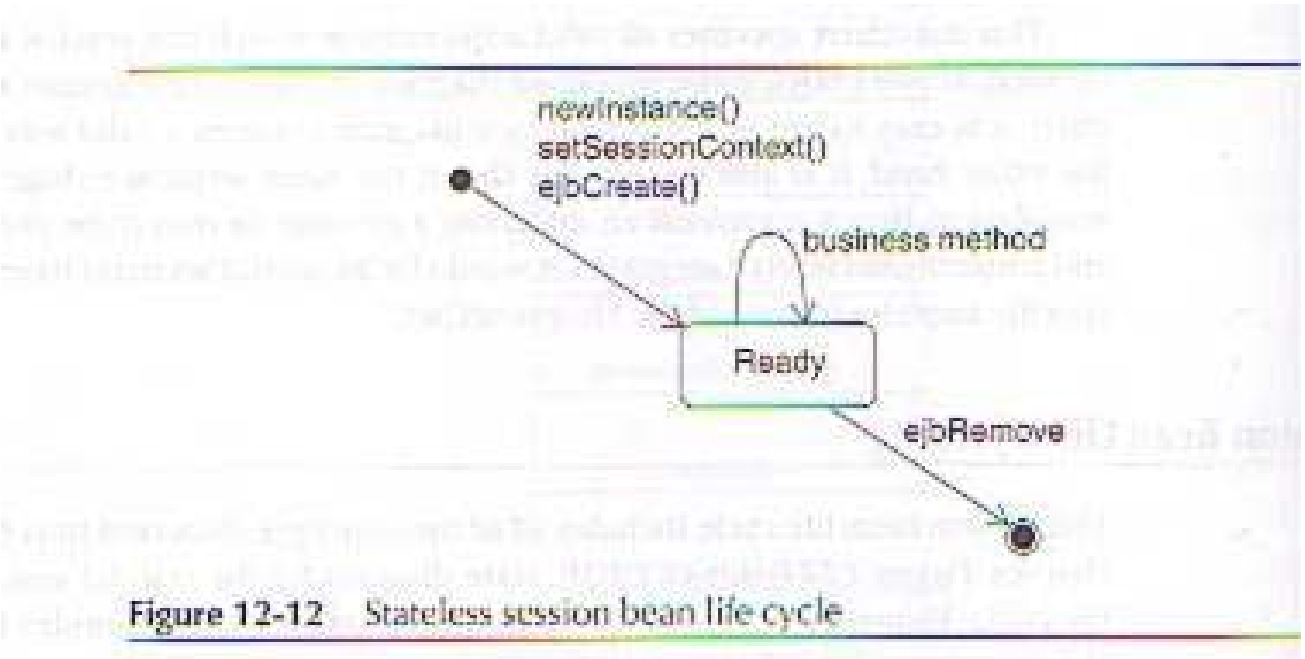
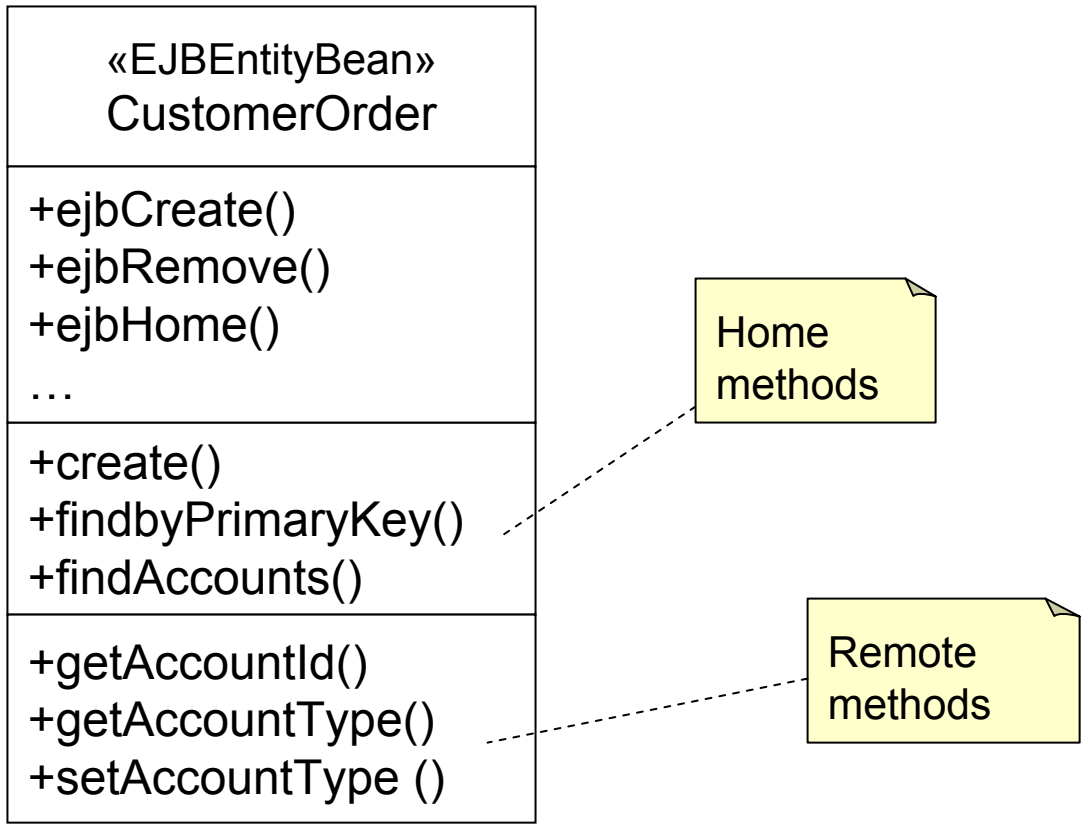


Figure 12-12 Stateless session bean life cycle

3.3.4.2 EJB - Entity Beans

Modélisation des Session Beans en UML



3.3.4.2 EJB - Entity Beans

Modélisation des Session Beans en UML

Vue interne

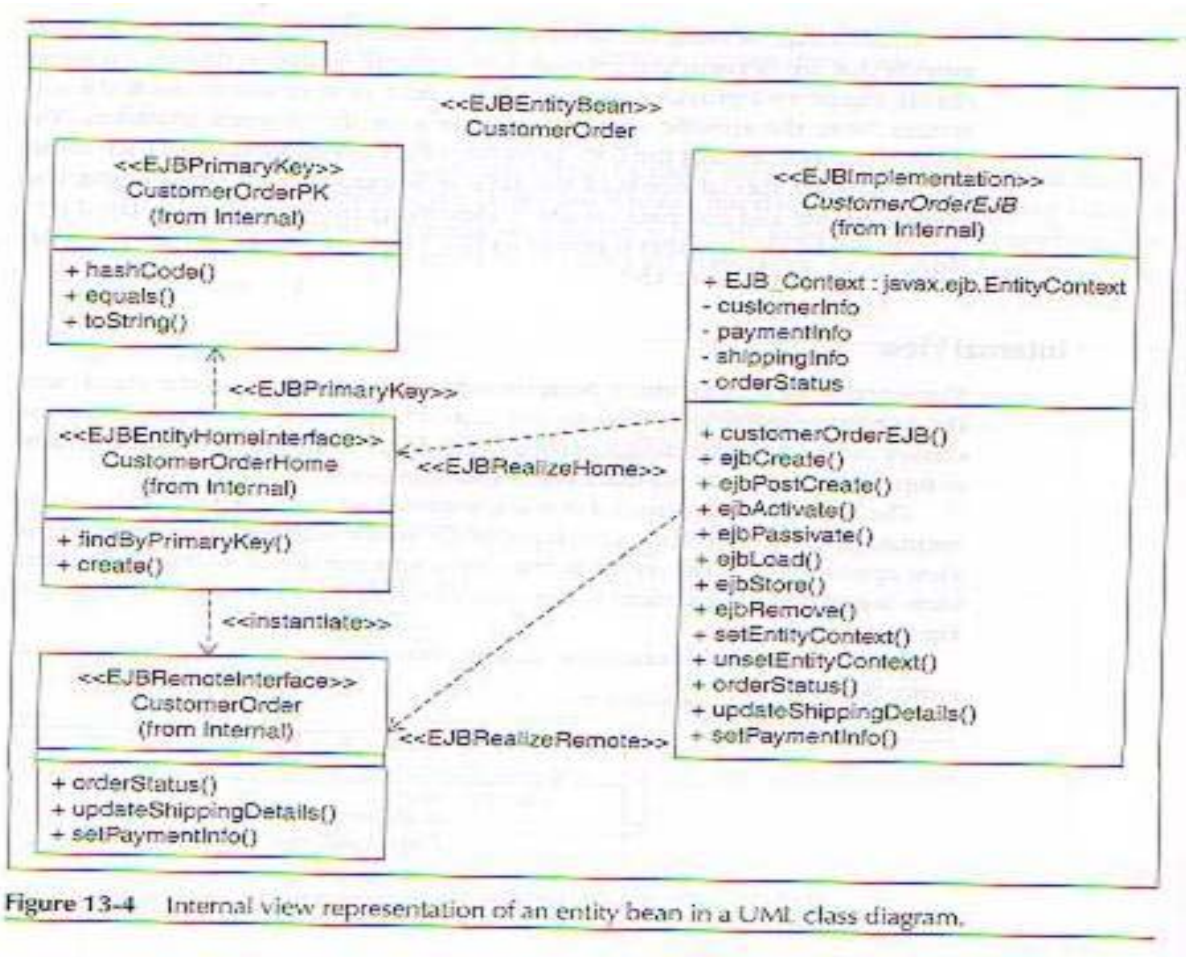
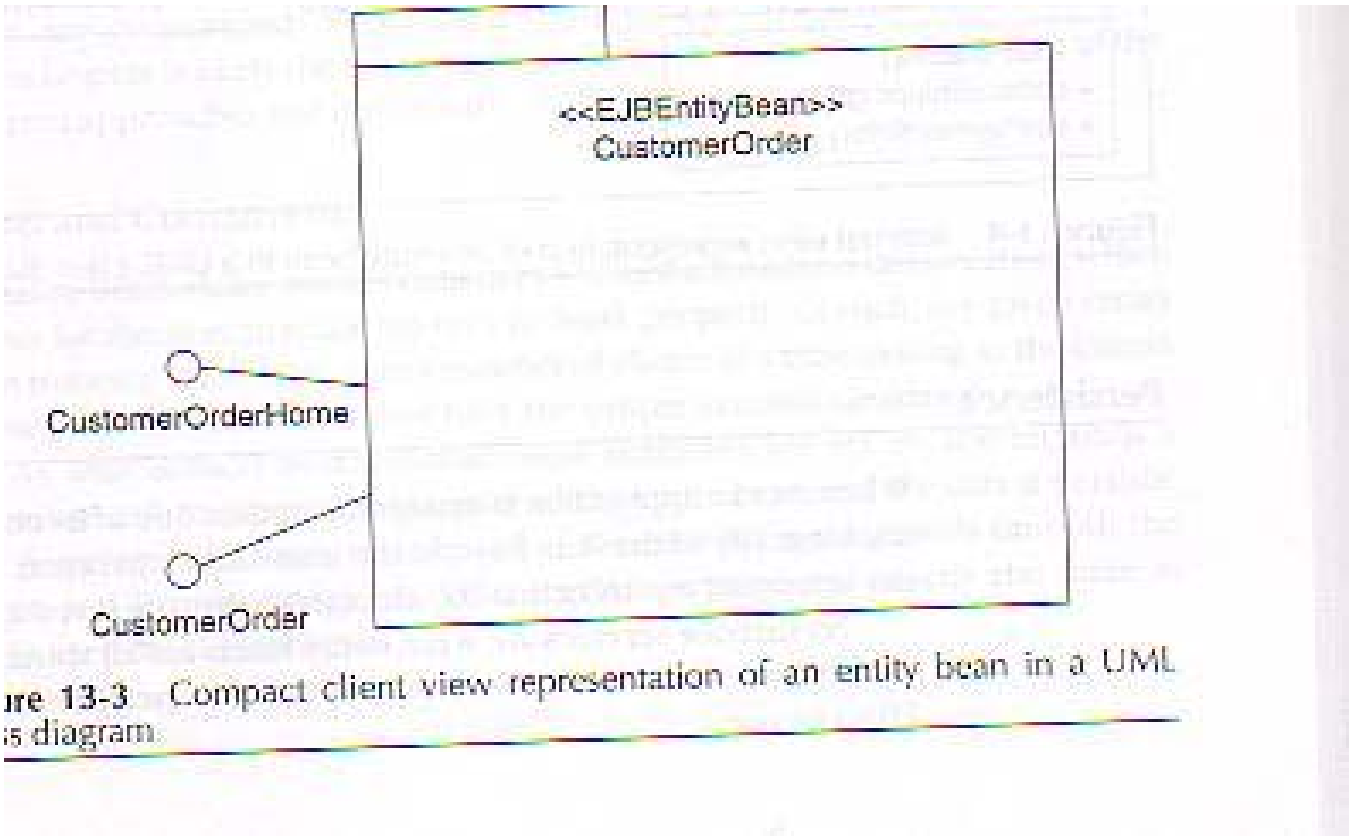


Figure 13-4 Internal view representation of an entity bean in a UML class diagram.

Entity Beans

Modélisation des Session Beans en UML

Vue externe



ure 13-3 Compact client view representation of an entity bean in a UML
s diagram