Applications d’Entreprise avec JEE

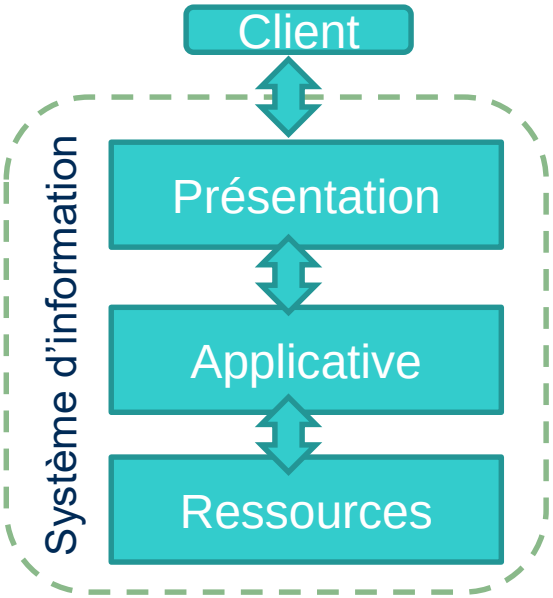
Karim Guennoun

Introduction aux Architectures Distribuées

Les trois niveaux d’abstraction

Les niveaux d'abstraction

- En règle générale, une application est constituée de 3 niveaux d'abstraction:
  - La couche présentation
    (presentation layer)
  - La couche application
    (application logic layer)
  - La couche gestion des ressources
    (ressource management layer)



La couche présentation (1/2)

- Tout système doit communiquer avec des entités externes:
  - Humains
  - Autres machines …
- Besoin de présenter convenablement l'information à ces entités:
  - Objectif: permettre de soumettre des opérations et d'avoir les réponses
- Les éléments qui permettent ce type de traitement appartiennent à la couche présentation

La couche présentation (2/2)

- Les éléments de cette couche ne doivent pas être confondus avec le client.
- Le client est l'utilisateur du système
- Exemples de module de présentation:
  - Interface graphique (GUI)
  - Pages/Formulaires HTML
  - Application mobile

La couche application

- Le système doit délivrer de l'information
- Généralement, il effectue des tâches de calcul sur des données
- Il implémente les opérations requises par le client à travers la couche présentation
- Les modules, programmes et composants qui réalisent cette implémentation constituent la couche applicative.

La couche application

- Les éléments sont appelés: services
- Exemple: service de retrait sur compte bancaire:
  - Prend la requête de retrait
  - Vérifie que le compte est assez approvisionné
  - Vérifie que le plafond de retrait n'est pas atteint
  - Donne l'autorisation de retrait pour la somme demandée
  - Met à jour le nouveau solde
- Autre appellation pour ce niveau: processus business, logique business, règles business, ou simplement: serveur.

La couche gestion de ressources

- Les systèmes d'information ont besoin de données sur lesquels travailler
- Les données sont stockées sur
  - Fichiers
  - Bases de données
  - ERPs
  - …
- La couche ressources correspond à ce type d'entités

Couche gestion de ressources

- Autre appellation: couche données
- Cela correspond au mécanismes de stockage persistant des données
- Dans des architectures plus complexes, la couche gestion de ressources peut aussi être un autre système d’information

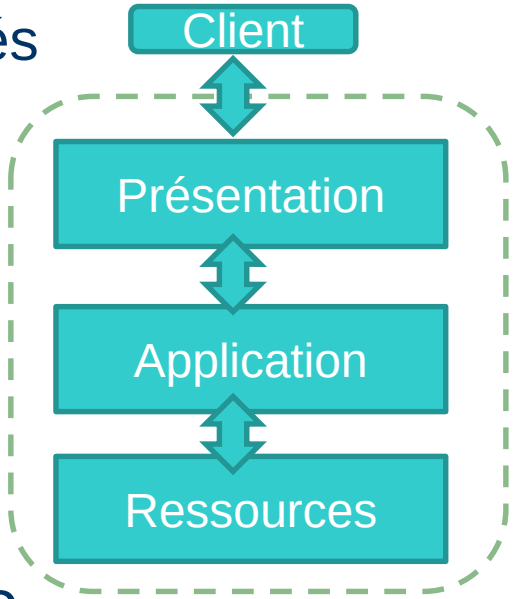
Des couches conceptuelles vers les tiers

- Les trois couches présentation, application, et ressources sont des couches conceptuelles pour séparer les fonctionnalités d'un système
- Dans les systèmes réels, ces couches peuvent être distribuées et combinées de différentes manières
- On parle alors de tiers
- Selon l'organisation considérée pour ces tiers, on obtient les modèles architecturaux:
  - 1-tier
  - 2-tier
  - 3-tier
  - N-tier

L’architecture 1-tier

Il était une fois l'architecture 1-tier

- Historiquement, au début
  - Gros serveurs et calculateurs déconnectés
  - Une interface réduite à un invite de commande
  - Problématique principale: utiliser efficacement la CPU
- Systèmes monolithiques
- Les trois couches sont dans le même tier
- Le système vu comme une boîte noire
- Pas d'interaction avec d'autres systèmes ni d'API



Architecture 1-tier

Avantages

- Possibilité de fusionner à souhait les différentes couches pour optimiser l'application
- Pas besoin de maintenir et de publier une interface,
- Aucune raison d'investir dans des transformations complexes de données pour la compatibilité
- Coût nul concernant le développement des clients et le déploiement de l'application.
- Sécurisation et diagnostiquabilité plus simples

Inconvénients

- Un code monolithique et rigide
- Efficace mais très couteux et difficile à maintenir
- Obsolète par rapport au matériel
- Absence des avantages liés à la distribution
- L’industrie du logiciel à pris le chemin opposé.

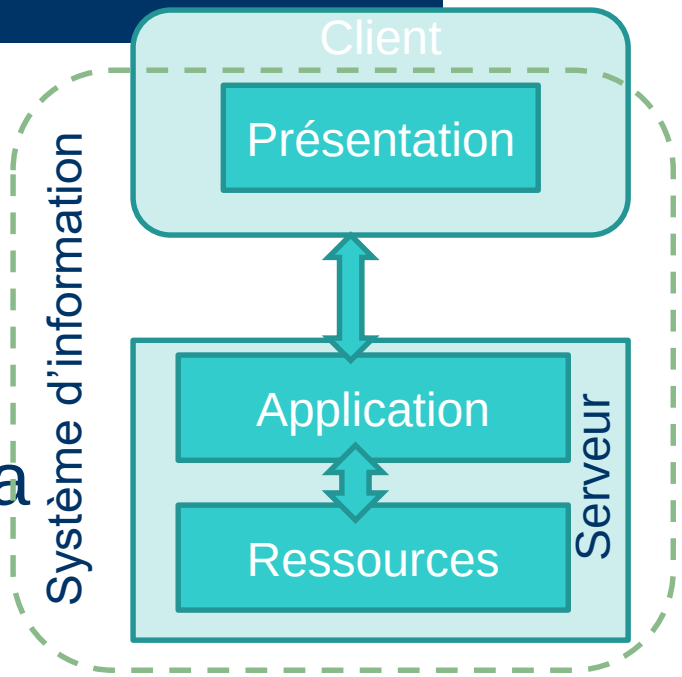
L’architecture 2-tier

L’architecture 2-tier

- Emergence due à l’apparition du PC
- Coexistence de machines moins puissantes (PC et stations de travail) avec de grosses machines (calculateurs et serveurs)
- Pour les designers:
  - Besoin de garder ensemble les couches gourmandes en ressources
  - La couche présentation est mise avec le client
- Avantages principaux:
  - Liés intrinsèquement à la distribution
  - Rendre possible la définition de plusieurs présentations pour la même application sans la rendre plus complexe
  - La couche présentation est déplacée dans le PC libérant de la puissance de calcul pour les deux autres couches

L’architecture 2-tier

- L’architecture 2-tier devient très populaire. On parle alors d’architecture Client-Serveur
- Le tier client correspond à la couche présentation
- Le tier serveur englobe les deux couches application et gestion des ressources



Développements associés à l'architecture Client-Serveur

- Les systèmes client-serveur ont permis plusieurs avancées dans les domaines du logiciel et du matériel avec une boucle vertueuse:
  - Avec l'augmentation des ressources dans les PCs et les stations de travail, la couche client est devenue de plus en plus sophistiquée.
  - La sophistication des clients a poussé vers une amélioration des performances dans les machines et dans les réseaux
- L'approche C-S est associée avec des développements cruciaux dans les systèmes distribués
  - La notion de RPC (Remote Procedure Call). Interaction à base d'appel de procédure.
  - Permet au concepteur de raisonner en terme d'interfaces publiées
  - La notion d'API.

Avantages sur le 1-tier

- La distribution
- Les couches application et ressources sont ensembles
  - L'exécution des opérations reste performante
- Indépendance du client et du serveur
  - Accès distants
  - Accès concurrents
  - Interopérabilité avec plusieurs plateformes
  - Possibilité de définir des couches présentation différentes pour différents clients à moindre coût

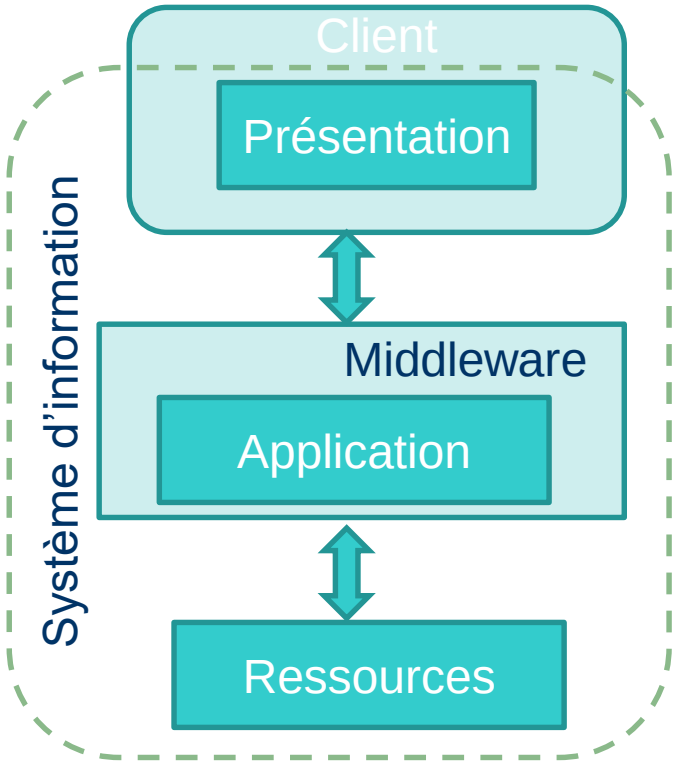
Inconvénients

- Apparition de problématiques nouvelles
  - Les problématiques de communication
  - Gestion des accès concurrents
  - Passage à l’échelle (scalability)
  - Sécurité
  - La recherche et publication
  - Les transactions

Architecture 3-tier

L’architecture 3-tier

- Une séparation claire entre les différentes couches abstraites:
  - La couche présentation réside chez le client
  - La couche application réside dans le tier du milieu
    - L’infrastructure qui supporte le développement de la logique business est appelée middleware
  - La couche gestion de ressources réside dans un troisième tier



Comparaison avec l'architecture 2-tier

- Dans l'architecture 2-tier la couche application et gestion des ressources sont co-localisées
  - Avantage: le coût de la communication est nul
  - Inconvénient: il faut une machine puissante pour exécuter les deux
- Pour l'architecture 3-tier, séparation des deux couches
  - Avantage:
    - Possibilité de les distribuer sur différentes machines
    - Augmentation de la performance
    - Les deux sont moins liées
    - Reutilisabilté
    - Maintenabilité
  - Inconvénient:
    - Complexité de l'architecture
    - Coût de communication additionnel entre les deux couches

Développements liés à l'architecture 3-tier

- L'apparition d'architectures 3-tiers a engendré des avancées:
  - Les gestionnaires de ressources ont dû s'adapter pour offrir des interfaces permettant la communication avec la couche application qui s'exécute dans le middleware
  - Apparition de standards de communication pour les gestionnaires de ressources pour un accès uniforme de la couche application
    - Java DataBase Connectivity: JDBC
    - Object Relational Mapping; ORMs
- L'architecture 2-tier a forcé l'apparition d'API pour la couche application alors que l'architecture 3-tier a provoqué l'apparition d'API pour la couche gestion des ressources

Développements liés à l'utilisation des middlewares (1/2)

- L'apparition des architectures 3-tier a induit l'apparition de middleware permettant:
  - Une intégration aisée de la logique métier
  - Des fonctionnalités pour la gestion des ressources:
    - Recherche/Publication
    - Communication
    - Accès concurrents/gestion des instances
    - Persistence
    - Garanties transactionnelles
    - …

Développements liés à l'utilisation des middlewares (2/2)

- Les concepteurs se concentrent sur la logique métier et profitent du support offert par le middleware pour le développement des interactions complexes
- La perte de performance liée à l'augmentation du coût de communication est généralement compensée par la possibilité de distribuer le tier du milieu sur plusieurs nœuds machine augmentant la scalabilité et la disponibilité de la communication

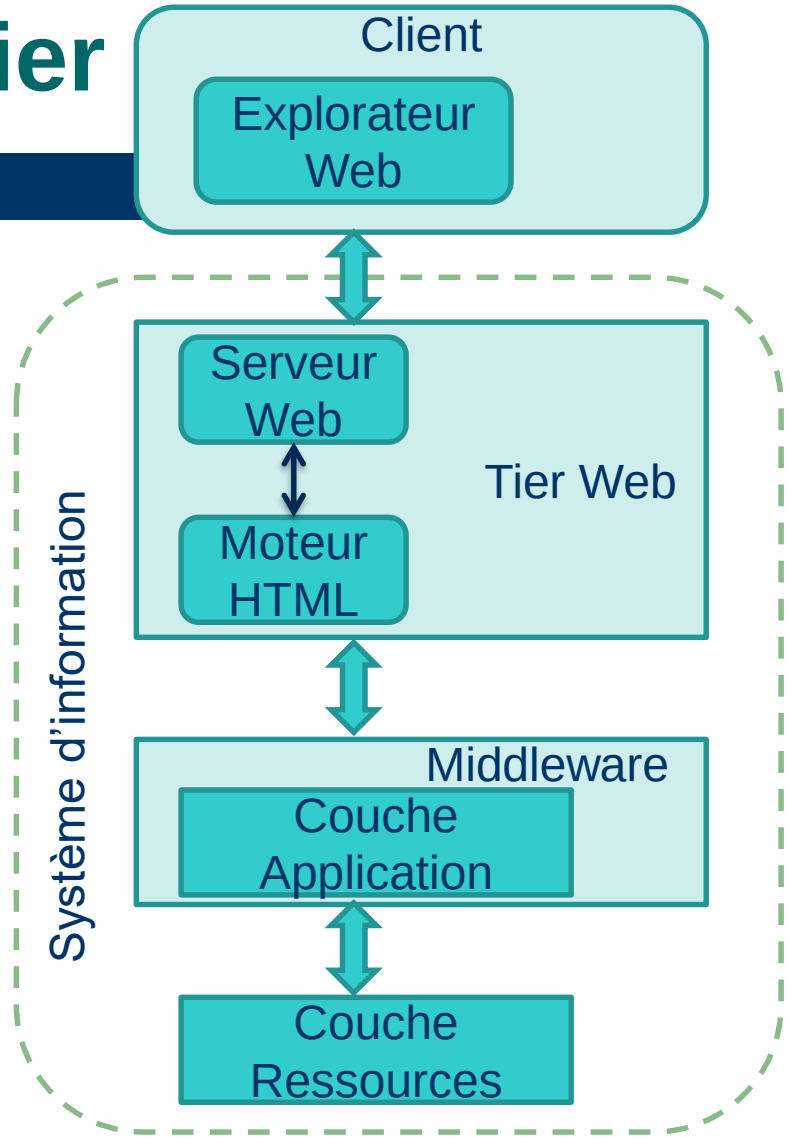
L’architecture N-tier

L’architecture N-tier

- Les architectures N-tier ne constituent pas réellement une évolution architectural par rapport à l’architecture 3-tier
- C’est une extension de ce modèle en considérant l’Internet comme un canal d’intéraction
- La majorité des systèmes construits actuellement

Une architecture N-tier

- La couche présentation est scindée en deux tiers
  - Un tier client comprenant un explorateur Web,
  - Un tier Web comprenant le serveur Web et le code qui prépare les pages HTML
- Les tiers Application et Gestion de données gardent la même sémantique



La distribution (Arrêt)

- En passant de l'architecture 1-tier vers la 2-tier, 3-tier, et N-tier, on assiste a une constante addition de tiers.
- Avec chaque tier,
  - l'architecture gagne en
    - Flexibilité
    - Fonctionnalité
    - Distribution
  - Introduit des coûts de communication additionnels entre les différents tiers
  - Introduit plus de complexité pour la gestion et la maintenance
- Il faut que le gain en flexibilité et en scalabilité compense les coûts de communication

Communication dans les systèmes d’information

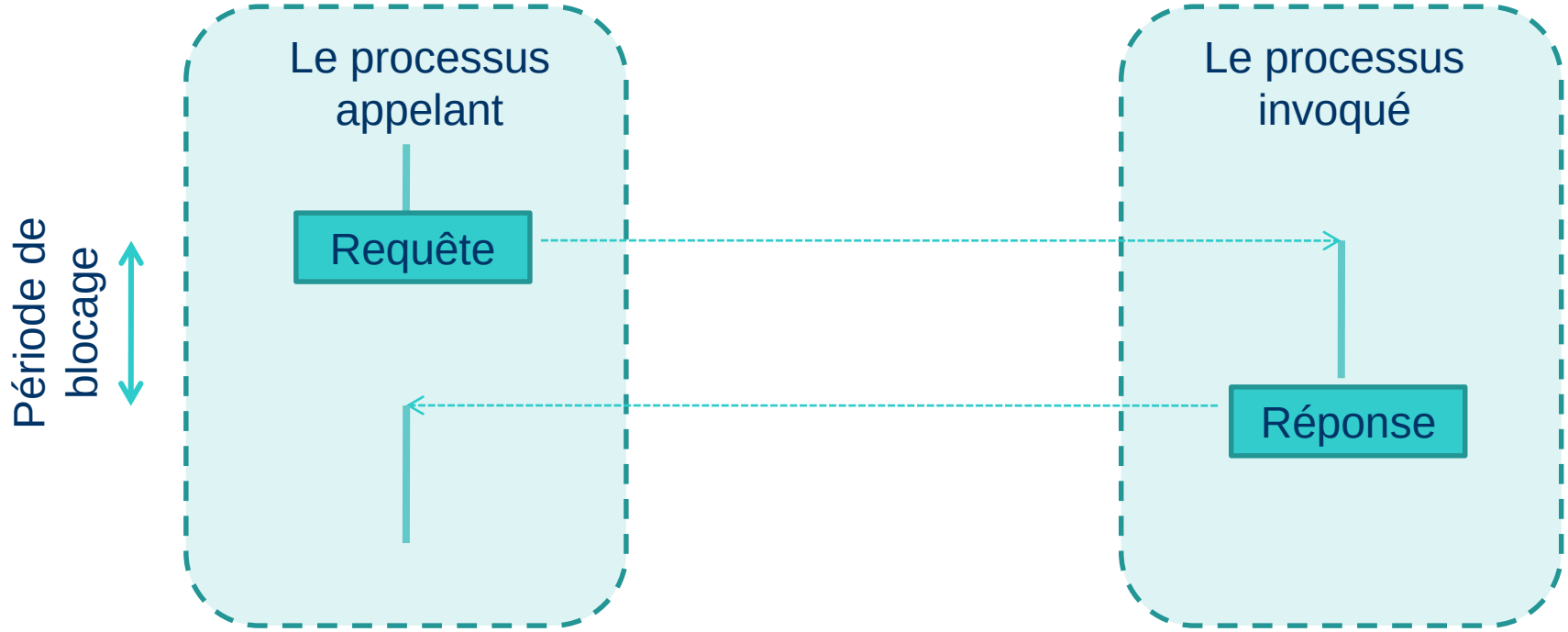
La communication dans les systèmes d'information

- Nous avons discuté comment les couches abstraites et les tiers peuvent être combinés et distribués
- Séparer le systèmes en plusieurs tiers implique l'implémentation des mécanismes de communication entre ces éléments
- La caractéristique dominante des différents mode d'interaction correspond aux choix synchrone ou asynchrone

Interactions Synchrones

Interaction synchrone

- Dans une interaction synchrone un processus qui appelle un autre doit attendre la réponse avant de continuer ses traitements.



Avantages (1/2)

- Attendre la réponse avant de continuer présente plusieurs avantages:
  - Simplifier la conception
  - L'état du processus appelant n'est pas altéré entre son appel et la réponse
  - Corrélation simple entre le code qui fait l'appel et le code qui traite la réponse (les deux bouts de code sont en séquence)
  - Les composants sont fortement liés pour chaque interaction. Plus de facilité pour le test, le débogage, et l'analyse de performance

Avantages (2/2)

- Au passage de l'architecture 1-tier vers 2-tier, la majorité des systèmes utilisent du synchrone pour la communication entre clients et serveur

- Au passage vers l'architecture 3-tier, la majorité des serveurs de données offrent une communication synchrone avec la couche application

Inconvénients

- Tous les avantages peuvent être aussi vus comme des inconvénients quand l'interaction n'est pas de type requête-réponse (e.g. one way)
- Perte de temps et de ressources calcul si le traitement côté serveur est long
- Le problème de performance augmente avec l'augmentation du nombre de tiers
- En terme de tolérance aux fautes, le processus appelant et le processus invoqué doivent être connectés et opérationnels lors de l'invocation. Ils doivent le rester tout au long de l'exécution de la requête
- Les procédures de maintenance doivent être faites offline

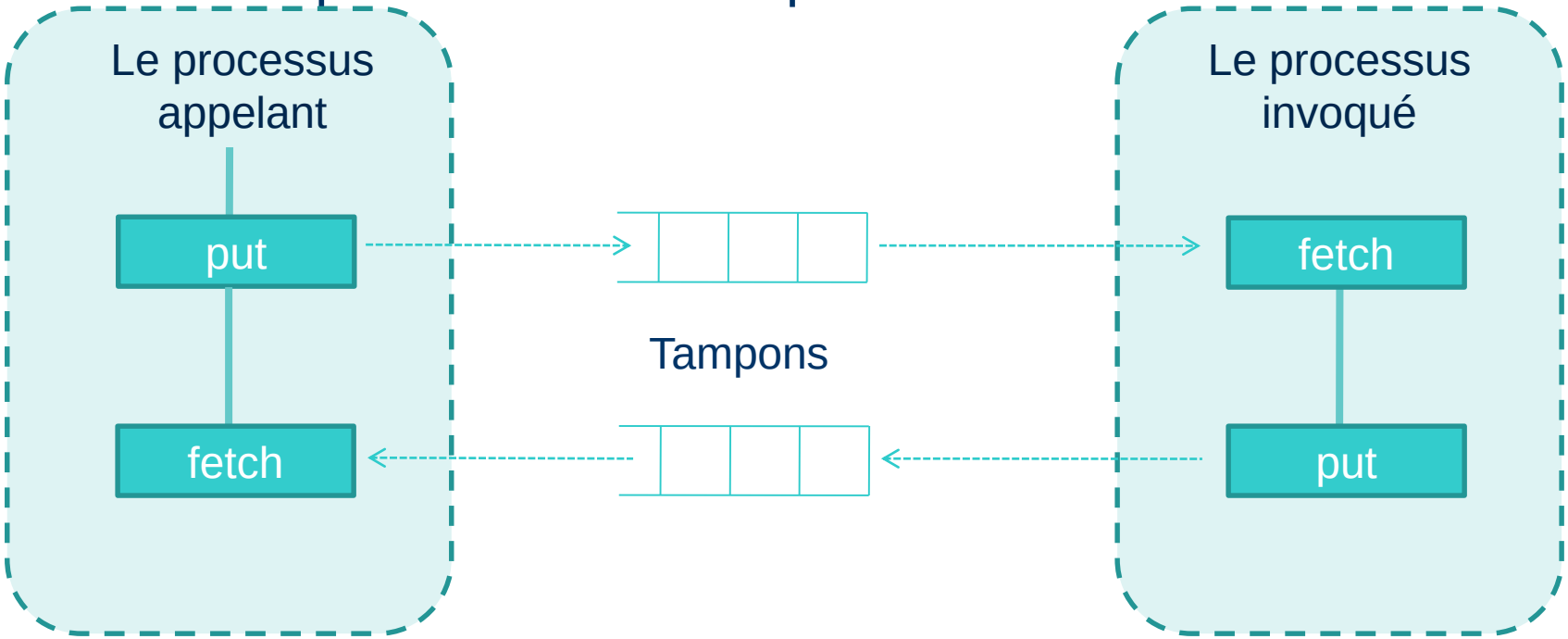
Interactions asynchrones

Les interactions asynchrones

- Quand il est nécessaire de travailler de manière interactive, le choix synchrone s'impose
- Dans plusieurs cas, cela n'est pas nécessaire
  - Impression
    - J'envois une demande d'impression
    - La demande est insérée dans la liste des taches
    - L'imprimante traite la tache quand elle est disponible
    - La machine est notifiée de la fin du traitement

Les interactions asynchrones

- Au lieu de faire une requête et attendre la réponse
  - Envoyer la requête
  - Vérifier plus tard si une réponse a été retournée



Avantages

- Suivre une approche non bloquante permet:
  - Au programme appelant de continuer à réaliser d'autres tâches pendant le traitement de sa requête
  - D'éliminer le traitement de la coordination entre les deux processus
- Réduction des problèmes dus
  - Au nombre de connections
  - À la dépendance entre composants
  - À la tolérance aux fautes
- Modèle adéquat dans certaines situations (style publisher-subscriber)
  - Un serveur dissémine l'information vers plusieurs clients
  - Différents clients s'intéressent à différents types d'information

Evolution des concepts de développement logiciel

Applications Procédurales

- E.g. C, Pascal, Ada
- L’univers est constitué principalement de procédures et de fonctions
- Interactions principalement via une interface IHM locale

L’orienté objet

- E.g. Java, C++, Eiffel
- L’univers est constitué principalement d’objets
  - Attributs
  - Méthodes
- L’orienté objet est plus une vision relative à l’organisation de l’implantation
  - La distribution et le déploiement ne sont pas pris en compte
- Concepts nouveaux:
  - Attributs (caractéristiques d’un objets)
  - Méthodes (actions possibles sur un objet)

L’orienté composant

- E.g. RMI, CORBA, EJB
- On s’intéresse maintenant à la structure en terme de
  - Décomposition du code exécutable stand-alone (composants)
  - Déploiement au niveau des environnements d’exécution
- Objectifs principaux, au sein d’une entité:
  - Réutilisation
  - Composition
  - Facilitation de développement
- Nouveaux Concepts:
  - Interfaces
  - Connexions
  - Middleware

L’orienté service

- E.g. Services Web
- Au-delà de l’orienté-composant
- Contexte: le web et les applications inter-entités
- Objectifs principaux
  - Développer le e-business
  - Publier-rechercher des services sur le web
  - Composition dynamique de services
- Nouveaux concepts
  - Registres de publications
  - Standards de
    - Description
    - Communication
    - Publication

Paradigmes de communication (arrêt)

- MOM
  - Orienté messages
- RPC
  - Orienté procédures
- RMI
  - Orienté méthodes

Ce qu’il faut retenir

Retour sur les thèmes abordés

- Trois couches conceptuelles sont identifiées
  - Présentation
  - Application
  - Gestion de ressources
- Quatre modèles architecturaux
  - 1-tier, 2-tier, 3-tier, N-tier
- Deux modèles de communication:
  - Client-Serveur
  - Publisher-Subscriber

La distribution, c'est Darwin!

- La distribution des couches conceptuelles a évolué en réponse à des évolutions au niveau matériel et réseau
  - Avec les gros serveur et calculateurs: l'architecture 1-tier
  - Avec les réseaux locaux et l'apparition des PCs et des stations de travail: l'architecture client-serveur
  - Avec la prolifération de l'information et l'augmentation de la bande passante: l'architecture 3-tier
  - Avec l'avènement d'Internet, une bande passante en augmentation constante, et l'essor du e-business: l'architecture N-tier

L’évolution de la programmation

- De programmes monolithique isolés
- Vers une structuration des applications en services distribués à travers le Web
- Développement d’outils et de standards
  - Conception
  - Développement
  - Déploiement
- De plus en plus de génération automatique de code
  - Stubs / Skeletons, classes de support
- Décharger un maximum le développeur pour se consacrer à la logique business

Introduction à la plateforme JEE

Le cadre (framework) java 2

- Le framework java comporte 3 éditions:
  - JSE: Java Standard Edition destinée au développement d'applications locales exécutables sur des ordinateurs personnels
  - JEE: Java Enterprise Edition destinée au développement d'applications distribuées impliquant des serveurs
  - JME: Java Micro Edition destinée au développement d'applications embarquées pour les terminaux mobiles.

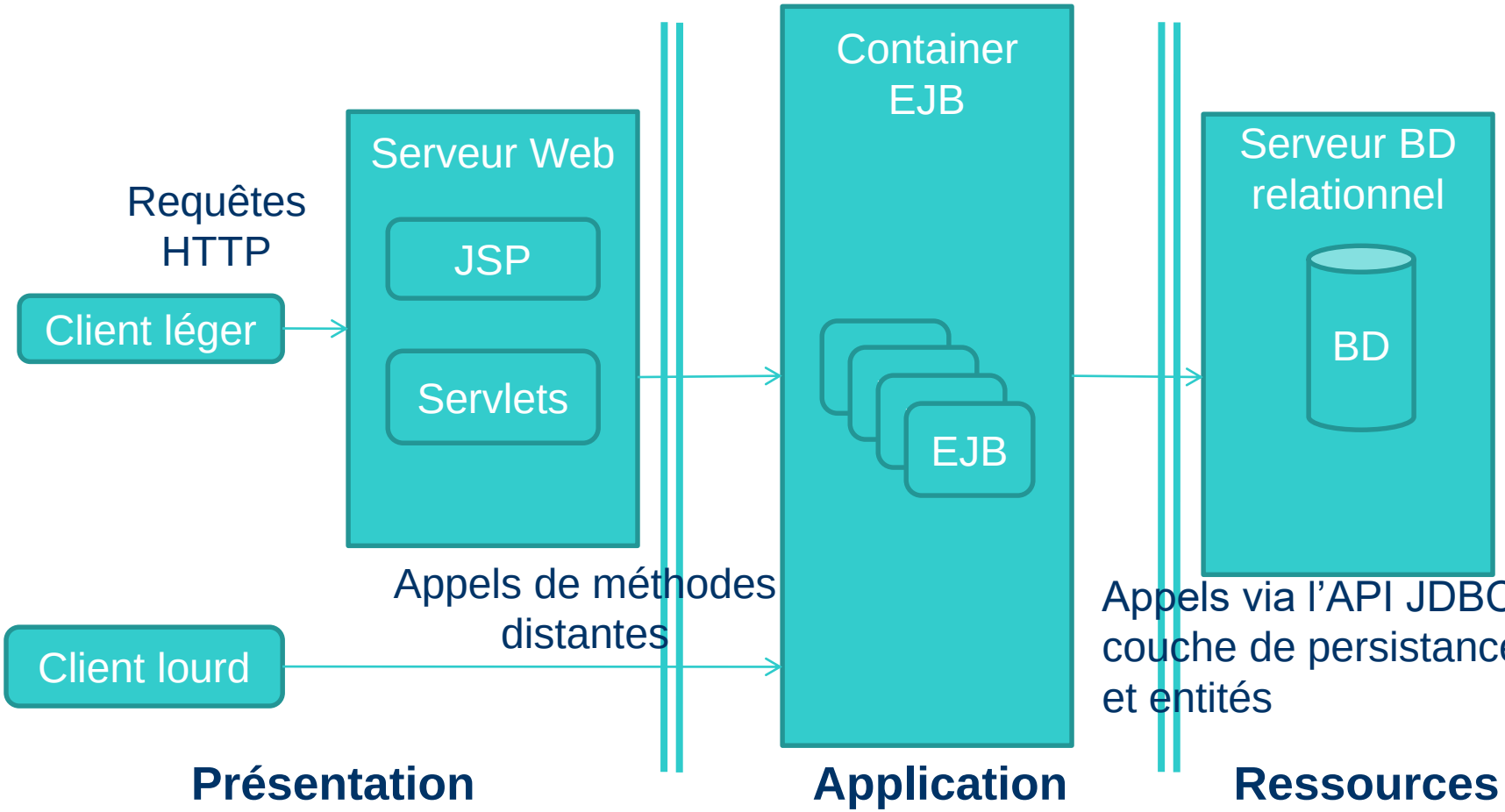
La plateforme JEE

- La plateforme JEE propose une approche **orientée-composants** pour la conception, le développement, le déploiement des applications distribuées
- JEE:
  - Repose sur un **modèle architectural N-tier**
  - Favorise la réutilisabilité des composants logiciels
  - Propose un modèle de gestion de la sécurité et des transactions
- Versions:
  - J2EE 1.2 (1999) /J2EE 1.3 (2001) / J2EE 1.4 (2003) / Java EE 5(2006) / Java EE 6 (2009) / Java EE 7 (2013) / Java EE 8 (2017) / Jakarta EE 8 (2019)

L’architecture JEE

- Une architecture JEE classique comprend les tiers suivants:
  - Le tier Client
  - Le tier Web
  - Le tier logique business
  - Le tier systèmes d’information

Exemple d'une architecture JEE



Les composants JEE (1/2)

- Les composants d'une architecture correspondent à des unités software "self-contained" possédant une interface définie
- Les applications JEE sont construites par assemblage de composants JEE
- La spécification JEE définit les composants suivants:
  - Les composants clients: clients applicatifs, clients web
  - Les composants web: Servlets et JSP
  - Les composants métier: EJB

Les composants JEE (2/2)

- Les composants JEE sont
  - Développés en langage java
  - Compilés comme tout programma java
- A la différence d'une classe JSE
  - Les composants sont connectés pour former l'application
  - Leurs implémentation et leurs interfaces sont vérifiées pour s'assurer de leur conformité à la spécification JEE
  - Les exécutables sont déployés sur le serveur JEE pour la production

Les clients JEE

Les clients applicatifs

- S'exécutent sur la machine du client
- Généralement, s'adressent directement au tier métier (e.g. des appels distants sur un composant EJB)
- Peuvent correspondre à
  - Une interface graphique Swing ou AWT
  - Une interface en ligne de commande
  - Une application mobile
  - Une autre application

Les clients Web

- Un client Web est constitué de deux parties
  - Les pages web dynamiques de type HTML ou XML générées par les composants web
  - Un browser qui interprète ces pages
- Généralement, un client Web est un client léger avec très peu de traitements

Les composants web

Les servlets

- Une servlet est un programme java déployé sur un serveur web
- Chargée automatiquement dans le serveur ou à la demande d'un client
- Une fois déployée elle reste en attente des requêtes clients
- Génère dynamiquement des données sous format de pages Web (format HTML ou XML)

Les JSPs

- Composants permettant de générer dynamiquement des pages web
- Le modèle JSP est dérivé du modèle servlet
- À l'appel d'un client, le serveur Web appelle le moteur JSP pour générer le code source, le compile pour générer l'exécutable de la servlet qui traite la requête

Les composants Java Bean

Les EJB

- La partie métier est implantée sous forme de composants EJB
- Les EJB réalisent les traitements, ils constituent le tier application
- Généralement, un EJB
  - reçoit l'information du tier client la traite et met à jour les données stockées dans le tier gestion des ressources
  - récupère l'information stockée dans le tier gestion des ressources la traite et renvoie le résultat de ses traitements au client

Les EJB

- Trois types d'EJB peuvent être utilisée:
  - Les session bean: non persistents
  - Les entity bean: persitents
  - Les messages-driven bean: combinaison de session bean et de JMS

Les composants gestion de ressources

La gestion de ressources

- Les composants correspondent à des entités logicielles mettant des données persistantes à la disposition des composants métier
- Les ressources traitées par une application JEE peuvent correspondre à:
  - une base de données relationnelle
  - un Entreprise Ressource Planning (ERP)
  - un système de fichiers
- Une couche de persistance peut être considérée afin de permettre
  - d'avoir une représentation objet de la BD
  - de gérer les aspects transactionnels et d'accès concurrents

Le middleware JEE

Les containers

- Un container constitue l'interface entre un composant et les fonctionnalités de bas niveau qu'offre la plateforme JEE tels que:
  - Le service JNDI: service de nommage
  - Le modèle security: gestion des accès et des authentifications
  - Le modèle transactionnel: gestion des transactions pour les accès aux données
- Avant l'exécution d'une application, ses composants doivent être assemblés en modules JEE et déployés dans un container

Les containers JEE

- Le container EJB
  - Gère l'exécution des EJBs.
  - Les EJBs et leurs containers s'exécutent au sein du serveur JEE
  - La majorité des serveurs JEE offrent des containers EJB intégrés
- Le container Web
  - Gère l'exécution des servlets et des JSPs
  - Les composants Web et leurs containers s'exécutent au sein du serveur JEE

Le serveur JEE

- Se charge de l'exécution des composants JEE
- Offre des containers Web et EJB
- Les serveurs JEE
  - Glassfish (Oracle)
  - JBOSS (Red Hat)
  - WebSphere Application Server (IBM)
  - Geronimo (Apache)

Présentation de l’API JEE

Quelques éléments de l'API

- Les EJBs (Enterprise Java Beans)
  - Composants métiers
  - Package: javax.ejb
- Les servlets
  - Composants Web
  - Package: javax.servlet
- Les JSP (Java Server Pages)
  - Composant Web
  - Package: javax.servlet.jsp

Quelques éléments de l'API

- JMS (Java Messaging Service)
  - Permet des échanges de messages asynchrones entre les composants
  - Package: javax.jms
- JavaMail
  - Permet l'envoie des notifications mail
  - Package: javax.mail
- JAXP (Java Api for Xml Processing)
  - Permet l'analyse et le parsing des fichiers XML
  - Package: javax.xml.parsers

Quelques éléments de l'API

- JAX-RPC (Java API for XML-Based RPC)
  - Permet des interactions RPC à base de messages SOAP
  - Package: javax.xml.rpc
- JTA (Java transaction API)
  - Permet la gestion des transactions
  - Package: javax.transaction
- JNDI
  - Implémente de service de nommage dans JEE
  - Package:javax.naming

Les Servlets

C’est quoi une servlet?

- Une servlet est un composant Web
- Déployée du côté serveur, sur un serveur Web
  - Reçoit une requête du client Web
  - Effectue des traitements
  - Renvoie une réponse

Fonction

- L’utilité première d’une servlet est la génération dynamique de pages html
- Généralement, les servlets utilisent le protocole http pour communiquer
  - Requêtes http
  - Réponses http
- Cependant, une servlet peut reposer sur n’importe quel protocole bâti selon le modèle requête/réponse

Avantages de l'utilisation des servlets

- La génération **dynamique** des pages html
- Les servlets sont codées en java
  - Portabilité
  - Accès aux APIs Java
    - Possibilité d'interaction avec
      - Les composants métier
      - Les composants gestion des ressources (BD, ERP, …)
  - Garbage collector

Fonctionnement d'une servlet

- La servlet construit dynamiquement la page HTML réponse et l'encapsule dans l'objet réponse transmis par le serveur en utilisant
  - Les valeurs transmises par la requête
  - Toute autre ressource disponible et accessible
    - Les données persistantes
    - Les composants métier
    - …
- Le serveur récupère la page HTML contenue dans l'objet réponse de la servlet
- Le transmet au client (browser)

Fonctionnement d'une servlet

- La servlet est déployée
  - En copiant son code exécutable dans un répertoire spécifique du serveur Web (pour Tomcat, c'est le répertoire webapps)
  - En définissant son descripteur de déploiement (pour Tomcat, web.xml)
- Le serveur Web charge la servlet automatiquement ou à la demande d'un client
- Une servlet offre un accès concurrent

Fonctionnement d'une servlet utilisant le protocole HTTP

- Le serveur reçoit une requête d'un navigateur spécifiant l'appel à une servlet
  - lien http correspondant à l'url définie pour la servlet dans le descripteur de déploiement
- A la première requête d'un client, le serveur instancie la servlet
- Si elle n'est pas arrêtée de manière active, la servlet reste en mémoire jusqu'à l'arrêt du serveur

Fonctionnement d'une servlet utilisant le protocole HTTP (arrêt)

- Pour chaque requête, le serveur crée un processus léger pour son traitement
  - Possibilité de traiter plusieurs requêtes à la fois
  - Limite spécifiée dans la configuration du serveur
    - Pour Tomcat, 150 par défaut (c.f. fichier XML conf/server.xml)
  - Le serveur crée
    - Un objet représentant la requête HTTP
    - Un objet qui va contenir la réponse HTTP
  - Envoie les deux objets à la servlet
  - Renvoie le résultat encapsulé dans la réponse au client

Les serveurs Web

Tomcat (apache)

Rôle du serveur Web

- Servir de plateforme d'exécution des servlets
- Gestion du cycle de vie
  - Chargement/rechargement
  - Démarrage/arrêt/retrait
  - Instanciation
- Gestion des interactions
  - Interceptions des requêtes destinées aux servlets
  - Création des objets requêtes et réponse
  - Retour des pages HTML résultat

Le serveur Web Tomcat

- Développé par la fondation Apache
- Implémente les spécifications servlet et jsp
- Utilise le container catalina (pour servlets)
- Compile les JSPs grâce au compilateur jasper
- Coyote est le connecteur http de tomcat
- Codé en Java
- Dernière version: 9.0.26 (Septembre 2019)

Installation de Tomcat

- Récupérer le code de Tomcat
- Le copier dans un répertoire (e.g. TomcatRep)
- Positionner correctement les variables d'environnements:
  - TOMCAT_HOME à TomcatRep
  - CATALINA_HOME à TomcatRep
  - JAVA_HOME au répertoire du JDK
- Pour pouvoir lancer Tomcat à partir d'un invite de commande mettre à jour la variable PATH: rajouter le chemin, TomcatRep\bin

Le contenu du code

- bin : Scripts et exécutables pour différentes tâches : démarrage, arrêt,
- lib : Classes communes que Catalina et les applications Web utilisent
- conf : Fichiers de configuration (format XML)
- logs : Journaux des applications Web et de Catalina ;
- webapps : Répertoire contenant les applications web déployées sur Tomcat
- work : Fichiers et répertoires temporaires.

Lancement de Tomcat

- Si le PATH mis à jour,
  - Ouvrir un invite de commande et lancer le programme via la commande: startup
- Sinon, aller sur le répertoire TomcatRep/bin et lancer l'exécutable startup
- Pour vérifier si Tomcat est correctement lancé,
  - Aller sur l'url: http://localhost:8080

Le manager de Tomcat

- url: http://localhost:8080/manager
- Permet de gérer le cycle de vie des composants Web
- Par défaut, il faut positionner un administrateur avec le rôle manager
- Aller sur le fichier TomcatRep/conf/tomcat-users.xml
- Rajouter un utilisateur avec une balise de la forme
  - <role rolename="manager-gui"/>
  - <user username="karim" password="karim" roles="manager-gui"/>

L’API Servlet

Les packages

- Deux packages
  - javax.servlet:
    - Permet de développer des servlets génériques indépendantes du protocole d'interaction
  - javax.servlet.http
    - Permet de développer des servlets interagissant avec le protocole HTTP

Le package javax.servlet

Les interfaces

- Servlet: définition de base d'une servlet
- ServletConfig: définition d'un objet pour configurer la servlet
- ServletContext: définit un objet contenant des informations sur le contexte d'exécution de la servlet
- ServletRequest: définit un objet contenant la requête du client
- ServletResponse: définit un objet contenant la réponse

Les classes

- GenericServlet: définit une servlet indépendante de tout protocole
- ServletInputStream: définit un flux permettant la lecture des donnes de la requête cliente
- ServletOutputStream: définit un flux permettant le renvoie de la réponse de la servlet

Les exceptions

- SevletException: Exception générale levée en cas de problème durant l'exécution de la servlet
- UnavailableException: Exception levée si la servlet n'est pas disponible

L’interface Servlet

- void init(ServletConfig conf):
  - Appelée une seule fois lors de l’instanciation de la servlet
- ServletConfig getServletConfig():
  - Renvoie l'objet ServletConfig passé à la méthode init
- void service (ServletRequest req,ServletResponse res):
  - A chaque requête du client cette méthode est exécutée
- void destroy():
  - Permet la destruction de la servlet
- String getServletInfo():
  - Renvoie les informations définies pour la servlet.

L’interface ServletRequest

- Méthodes
  - ServletInputStream getInputStream()
  - BufferedReader getReader()
  - Permettent d'obtenir un flux de lecture sur les données de la requête

L’interface ServletResponse

- Méthodes
  - SetContentType: Permet de préciser le type de la réponse
    - /text/html pour une page html
  - ServletOutputStream getOutputStream()
    - Permet d'obtenir un flux pour envoyer la réponse
  - PrintWriter getWriter()
    - Permet d'obtenir un flux d’écriture pour envoyer la réponse

Démonstration d’implémentation

Implémentation d'une servlet

- Une servlet est codée comme n'importe quelle classe java
- Implémente l'interface javax.servlet.Servlet
- Définir, entre autres, la méthode
  - service (ServletRequest req, ServletResponse res) permettant de récupérer la requête et de retourner le résultat

Une servlet qui dit bonjour

- Dans un fichier MaPremiereServlet.java:

```
import java.io.*;
import javax.servlet.*;

public class MaPremiereServlet implements
    Servlet {
private ServletConfig cfg;
```

Implémentation des méthodes de l'interface Servlet

```
public void init(ServletConfig config)  throws ServletException
{cfg = config;}

public ServletConfig getServletConfig()
{ return cfg; }

public String getServletInfo()
{return("C'est ma première Servlet");}

public void destroy() { }
```

La méthode service

```
public void service (ServletRequest req, ServletResponse res )
throws ServletException, IOException {
res.setContentType( "text/html" );
PrintWriter out = res.getWriter();
out.println( "<HTML>" );
out.println( "<HEAD>");
out.println( "</HEAD>" );
out.println( "<BODY>" );
out.println( "<H1 align=\"center\">Je suis content ma premiere Servlet me
    repond:</H1>" );
out.println( "<H2>Bonjour</H2>" );
out.println( "</BODY>" );
out.println( "</HTML>" );
out.close();}}
```

Compilation de la servlet

1. S’assurer que les variables d’environnement relatifs à java sont bien positionnés
    1. JAVA_HOME=…
    2. JDK_HOME=…
    3. PATH=%PATH%;…\TomcatRep\bin
2. S’assurer que le package JEE est inclus dans le classpath
    1. classpath=.;C:\java\lib\tools.jar;C:\java\jre\lib\rt.jar
    2. classpath=%classpath%;java\lib\Javaee.jar
3. Compiler la classe MaPremiereServlet
    1. javac MaPremièreServlet.java

Arborescence du code exécutable

- Sous le répertoire **webapps** de Tomcat
  - Un répertoire correspondant à l'application Web, sous lequel se trouver:
    - Un répertoire **WEB-INF** qui contient au moins:
      - Un répertoire **classes**
        - Les exécutables
      - Un fichier **web.xml**
        - Le descripteur de déploiement

Descripteur de déploiement

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
        <servlet>
                <servlet-name>Toto</servlet-name>
                <servlet-class>HelloWorldExample</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>Toto</servlet-name>
                <url-pattern>/Toto</url-pattern>
        </servlet-mapping>
</web-app>
```

Exécution

- Lancer Tomcat
- Ouvrir le bowser
- Envoyer une requête à la servlet sur l'url:
  - http://localhost:8080/NomApplicationWeb/URL_Web.xml

Les servlets HTTP

Le package

La classe HttpServlet

- L'API fournit une classe qui encapsule une servlet utilisant le protocole http
- La classe HttpServlet
  - Hérite de la classe GenericServlet qui implémente l'interface Servlet
  - Redéfinie toutes les méthodes nécessaires pour fournir un niveau d'abstraction permettant de développer facilement des servlets avec le protocole http

Interaction avec les servlets http

- La méthode service héritée de HttpServlet appelle l'une ou l'autre de ces méthodes en fonction du type de la requête http :
  - une requête GET : c'est une requête qui permet au client de demander une ressource
  - une requête POST : c'est une requête qui permet au client d'envoyer des informations issues par exemple d'un formulaire
  - Des requêtes PUT, DELETE…
- En pratique, la méthode service analyse HttpServletRequest et exécute doGet, doPost, doPut… selon le type de la requête

Le package javax.servlet.http

- Les interfaces
  - HttpServletRequest: Hérite de ServletRequest et définit un objet contenant une requête selon le protocole http
  - HttpServletResponse: Hérite de ServletResponse et définit un objet contenant la réponse de la servlet
  - HttpSession: Définit un objet qui représente une session d'interaction avec la servlet

Démonstration d’implémentation

Implémentation d'une servlet http

- Hérite de la classe HttpServlet
- Définir les méthodes
  - doGet(HttpServletRequest req, HttpServletResponse res) permettant de renvoyer la réponse pour une requête http Get
  - doPost(HttpServletRequest req, HttpServletResponse res) permettant de renvoyer la réponse pour une requête http Post

Une servlet http qui dit bonjour

- Dans un fichier MaDeuxiemeServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaDeuxiemeServlet extends
    HttpServlet {
```

Implémentation de la méthode Get

```
public void doGet (HttpServletRequest req, HttpServletResponse res )
throws ServletException, IOException {
res.setContentType( "text/html" );
PrintWriter out = res.getWriter();
out.println( "<HTML>" );
out.println( "<HEAD>");
out.println( "</HEAD>" );
out.println( "<BODY>" );
out.println( "<H1 align=\"center\">Je suis content, ma Servlet HTTP me répond</H1>" );
out.println( "<H2>Bonjour</H2>" );
out.println( "</BODY>" );
out.println( "</HTML>" );
out.close();
}
```

Implémentation de la méthode Post

```
public void doPost(HttpServletRequest
    request,HttpServletResponse response)
throws ServletException,IOException
{
this.doGet(request, response);
}
}
```

Déploiement de la servlet sur Tomcat5.5

- Déployer la servlet
  - Copier le fichier MaDeuxiemeServlet.class dans le répertoire
    - CheminTomcat\webapps\...\WEB-INF\classes
  - Modifier le descripteur de déploiement, fichier :
    - CheminTomcat\webapps\...\WEB-INF\web.xml

Modification du descripteur web.xml

- Rajouter les balises
  - <servlet>
    <servlet-name>MaDeuxiemeServlet</servlet-name>
    <servlet-class>MaDeuxiemeServlet</servlet-class>
    </servlet>
  - <servlet-mapping>
    <servlet-name>MaDeuxiemeServlet</servlet-name>
    <url-pattern>/MaDeuxiemeServlet</url-pattern>
    </servlet-mapping>

Utilisation des formulaires

Une servlet http qui dit bonjour en réponse à un formulaire

- Dans un fichier MaTroisiemeServlet.java:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaTroisiemeServlet extends
    HttpServlet {
```

Implémentation de la méthode Get

```
public void doGet(HttpServletRequest request, HttpServletResponse
    response)
throws IOException, ServletException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Bonjour tout le monde</title>");
out.println("</head>");
```

Implémentation de la méthode Get

```
…
out.println("<body>");
out.println("<FORM
    ACTION=\"http://localhost:8080/UneAppli/MaTroisiemeServlet\"
    METHOD=\"POST\">");
out.println("<H3> Votre nom s il vous plaît:");
out.println("<INPUT NAME=\"NOM\">");
out.println("<INPUT TYPE=\"SUBMIT\" Value=\"envoyer\">");
out.println("</FORM>");
out.println("</body>");
out.println("</html>");
}
```

Implémentation de la méthode Post

```
public void doPost(HttpServletRequest request,HttpServletResponse
    response)
throws ServletException,IOException {
response.setContentType( "text/html" );
PrintWriter out = response.getWriter();
out.println( "<HTML>" );
out.println( "<HEAD>");
out.println( "</HEAD>" );
```

Implémentation de la méthode Post

```
…
out.println( "<BODY>" );
out.println( "<H1 align=\"center\">Je suis content ma Troisième
    Servlet me répond personellement!</H1>" );
out.println( "<H2>Bonjour "+request.getParameter("NOM")+"</H2>" );
out.println( "</BODY>" );
out.println( "</HTML>" );
out.close();
}
}
```

Déploiement de la servlet sur Tomcat5.5

- Déployer la servlet
  - Copier le fichier MaTroisiemeServlet.class dans le répertoire
    - CheminTomcat\webapps\examples\WEB-INF\classes
  - Modifier le descripteur de déploiement, fichier :
    - CheminTomcat\webapps\examples\WEB-INF\web.xml

Modification du descripteur

- Rajouter l'élément
  - <servlet>
    <servlet-name>MaTroisiemeServlet</servlet-name>
    <servlet-class>MaTroisiemeServlet</servlet-class>
    </servlet>
  - <servlet-mapping>
    <servlet-name>MaTroisiemeServlet</servlet-name>
    <url-pattern>/MaTroisiemeServlet</url-pattern>
    </servlet-mapping>

Un petit exercice:

- Implémenter deux servlets
  - Une servlet Ping
    - Avec un boutton ping pour le doGet
  - Une servlet Pong
    - Avec un boutton pong pour le doGet
  - Le boutton ping de la servlet Ping renvoie vers la servlet Pong
  - Le boutton pong de la servlet Pong renvoie vers la servlet Ping
- Utiliser des servlets sur les différentes machines en fournissant les adresses IP des collègues

Un deuxième exo

- Un servlet Accueil permettant de choisir une opération de
  - Soustraction
  - Ou d'addition
- Une servlet Addition permettant de
  - Saisir deux valeurs entières
  - D'afficher leur somme
  - Et de fournir un bouton retour
- Une Servlet Soustraction avec un fonctionnement similaire à la servlet Addition

Les Java Server Pages(JSP)

La technologie JSP

- Les JSP constituent une technologie Java dont l'objectif est de permettre la génération dynamique des pages web
- L'atout majeur est de permettre la séparation des traitements et de la présentation au niveau du code de développement
- Généralement,
  - Présentation sous forme de code HTML
  - Traitement sous forme de code Java insérée dans des tags spéciaux

Fonctionnement

- Les JSPs constituent « un script » pour le développement de servlets
- Au premier appel de la JSP, le serveur web
  - Génère la servlet correspondante
    - Le code HTML est repris dans la servlet
    - Le code Java est inséré dans le code de la servlet
  - Compile la servlet
  - Charge la servlet et l'exécute
- Pour les appels suivants, la servlet est directement exécutée, donc plus rapide

JSP Vs Servlet

- JSPs et Servlets possèdent beaucoup de points communs (une JSP est traduite en une servlet!)
  - Les JSPs permettent un développement moins pénible (pas besoin de flux pour la génération de la page html) pour la partie présentation
  - Pas de phase de compilation active
  - Facilite la mise à jour et la modification
  - Pas de descripteur de déploiement
- Code HTML versus Code Java
  - Code HTML : JSP
  - Code Java: Servlet

Contenu d'une JSP

- Une JSP est un fichier qui possède, par convention, l'extension jsp
- Un fichier JSP est constitué de:
  - de tags HTML
  - de tags JSP
    - Directives
    - Scripting

Développons une JSP: La démo d'illustration

Une JSP correspondant à une page HTML (uniquement des tags HTML)

- Ecrivons une JSP renvoyons le même code de la première servlet
- Dans un fichier JSP_statique.jsp introduire le code HTML suivant:

```
<HTML>
<HEAD>
<META content="charset=ISO-8859-4">
</HEAD>
<BODY>
<H1 align="center">On est tous contents</H1>
<H2>Nous écrivons notre première JSP</H2>
</BODY>
</HTML>
```

Déploiement

- Enregistrer le fichier jsp sous webapps, par exemple sous:
  - /MesPremieresJSPs/JSP1.jsp
- Sur le navigateur accéder à la jsp par l'url:
  - http://localhost:8080/MesPremieresJSPs/JSP1.jsp

Regardons ce qui se passe au niveau de Tomcat

- Le code d'interprétation du script JSP vers le servlet est généré dans le répertoire /tomcat/work/…/MesPremieresJSPs
- L'exécutable est aussi généré dans ce répertoire
- Modifier le code de la JSP (e.g. supprimer une ligne)
- Recharger la jsp
- Revérifier le code de la servlet générée
- Appeler la servlet
- Re-revérifier le code

Plus loin

Les tags jsp

- En plus des tags HTML, il existe deux types de tags jsp :
  - Les tags de directives : permettent de contrôler la structure de la servlet générée
  - Les tags de scripting: permettent d'insérer du code Java dans la servlet

Les tags de directives

- Les tags de directives possèdent la syntaxe:
  - <%@ directive attribut="valeur" ... %>
- Les spécifications des JSP les directives :
  - page : permet de définir des options de configuration
  - include : permet d'inclure des fichiers statiques dans la JSP avant la génération de la servlet

La directive page (1/2)

- Possède les attributs:
  - contentType="text/html;charset=ISO−8859−1"
    - permet de préciser le type des données générées.
    - équivalente à response.setContentType(..);
  - isErrorPage="true|false"
    - Cette option permet de préciser si la JSP génère une page d'erreur.
  - errorPage="URL"
    - Cette option permet de préciser la JSP appelée au cas où une exception est levée

La directive page (2/2)

- extends="classe mere"
  - Permet de spécifier une classe mère pour la servlet générée
- import= "{package.class}":
  - Permet de spécifier les packages à importer pour le code java inclus dans la jsp
- info="Les infos":
  - permet de spécifier la chaîne de caractères renvoyée par la méthode getServletInfo
- isThreadSafe="true|false":
  - permet de spécifier au serveur web le mode d'interaction de la servlet générée (un seul traitement de requête à la fois (safe=false) ou plusieurs en parallèle(safe=true))
- …

La directive include

- Syntaxe:
  - <%@ include file="chemin du fichier" %>
- Permet d'inclure un fichier dans le code source de la jsp
- Ce fichier peut correspondre à
  - Du code HTML (ne contenant pas les balises HTML et BODY)
  - Un fragment de jsp
  - Du code java
- Permet de réutiliser le même fragment
- Avant génération du code de la servlet, le fragment est inclus

Une mini démo

- Prenons le fichier destinataires.htm suivant:

```
<ul>
<h3>
<li> Professeur</li>
<li> Etudiants SIG</li>
<li> Etudiants GI </li>
</h3>
</ul>
```

- Dans la JSP inclure ce fichier avec la directive:
  - <%@ include file="destinataires.htm" %>

Les tags de scripting

- Ce type de tags permet d'insérer du code java dans la servlet générée à partir de la JSP.
- trois tags scripting
  - le tag de déclaration : Permet la déclaration de variables et de méthodes
  - le tag d'expression : évalue une expression et insère le résultat dans la page web générée.
  - le tag de scriptlets : permet d’en inclure le contenu dans la méthode service() de la servlet.

Les tags de déclaration

- Syntaxe :
  - <%! Declaration %>
- Permet de déclarer des variables et des méthodes
- Ne génère pas de code HTML dans la JSP

Les tags d’expression

- Syntaxe:
  - <%= expression à évaluer%>
- L'expression est évaluée et convertie en chaîne avec un appel à la méthode toString()
- Le résultat est inclus dans la page générée

Exemple

- Déclarons deux variables i et j et affichons leur somme
- insérer le code:

```
<%! int i1= 5; int i2=10;%>
<%! int somme(int i,int j)
{
return(i+j);
}%>
<%! int resultat=somme(i1,i2);%>
<h2> la somme de <%=i1%> et de <%=i2%> est: <%=resultat%>
    </h2>
```

Les tags scriplets

- Syntaxe:
  - <% code java %>
- Permettent d'insérer du code java dans la méthode service de la servlet générée
- Le code doit contenir exclusivement du java
  - Pas de balises HTML ou jsp

Exemple

- Rajouter le code suivant:

```
<%! int fact=1;%>
<% for(int i=1;i<=10;i++)
{%>
<% fact*=i;%>
<H3> fact de <%=i%> est égal à <%=fact%>
   </H3>
<%}%>
```

Les commentaires

- Pour les JSP, il est possible d'inclure deux types de commentaires
  - Les commentaires de la JSP
    - Syntaxe: <%-- commentaire --%>
  - Les commentaires du code HTML généré
    - Syntaxe: <!-- commentaire -->
- Introduire ces deux types de commentaires dans le code de votre JSP
- Conclure

Petit Exo

- Ecrire une JSP qui demande, sur la ligne de commande après affichage par System.out.println, deux entiers a et b et renvoie sur une page html le résultat de leur division.

- Introduire une JSP d'erreur pour prendre en considération la division par 0