Introduction à Java EE

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 12 septembre 2016

LAMSADE
UMR CNRS 7243
laboratoire d'analyse et modélisation de systèmes pour l'aide à la décision

DAUPHINE
UNIVERSITÉ PARIS

PSL
RESEARCH
UNIVERSITY

Le terme Java

Terme *Java* adopté en 1995 ("as an example of yet another name that would never work") (source: Java World)

Le terme Java

Terme *Java* adopté en 1995 ("as an example of yet another name that would never work") (source: Java World)

A jar full of Java beans, please

- JAR File : introduits à la version 1.1. Une collection de fichiers `.class`.
- Java Bean (aussi version 1.1). (specs) : un composant logiciel pour assemblage (par exemple, un bouton AWT, une feuille de calcul à placer dans un document).

Fun fact

Voyons le nombre magique des fichiers `.class`...

Java EE

- Java EE ?

Java EE

- Java EE ? Java Platform, Enterprise Edition

Java EE

- Java EE ? Java Platform, Enterprise Edition
- JCP ?

Java EE

- Java EE ? Java Platform, Enterprise Edition
- JCP ? Java Community Process

Java EE

- Java EE ? Java Platform, Enterprise Edition
- JCP ? Java Community Process
- API ?

Java EE

- Java EE ? Java Platform, Enterprise Edition
- JCP ? Java Community Process
- API ? Application Programming Interface

Java EE

- Java EE ? Java Platform, Enterprise Edition
- JCP ? Java Community Process
- API ? Application Programming Interface

Java EE

- technologies
- Spécifications, dont API
- Implémentation de référence
- Version actuelle : 7

Java EE : Processus

- Java EE fortement appuyée sur standards ouverts
- Standards du W3C / IETF ?

Java EE : Processus

- Java EE fortement appuyée sur standards ouverts
- Standards du W3C / IETF ? HTTP, HTML, XML, WSDL, ...

Java EE : Processus

- Java EE fortement appuyée sur standards ouverts
- Standards du W3C / IETF ? HTTP, HTML, XML, WSDL, ...
- JCP : implication de « la communauté » pour standards Java
- Tentions entre standard ouvert et contrôle ! (2010, Apache quitte le comité JCP ; Doug Lea également, en faveur de OpenJDK...)

Standards JCP

- Implication de « la communauté » pour standards Java
- JCP ?

Standards JCP

- Implication de « la communauté » pour standards Java
- JCP ? Java Community Process

Standards JCP

- Implication de « la communauté » pour standards Java
- JCP ? Java Community Process
- Définit les JSR : standards utilisés en Java SE ou Java EE
- Spécifications tiennent compte de nombreux avis d'horizons divers
- Exemples ?

Standards JCP

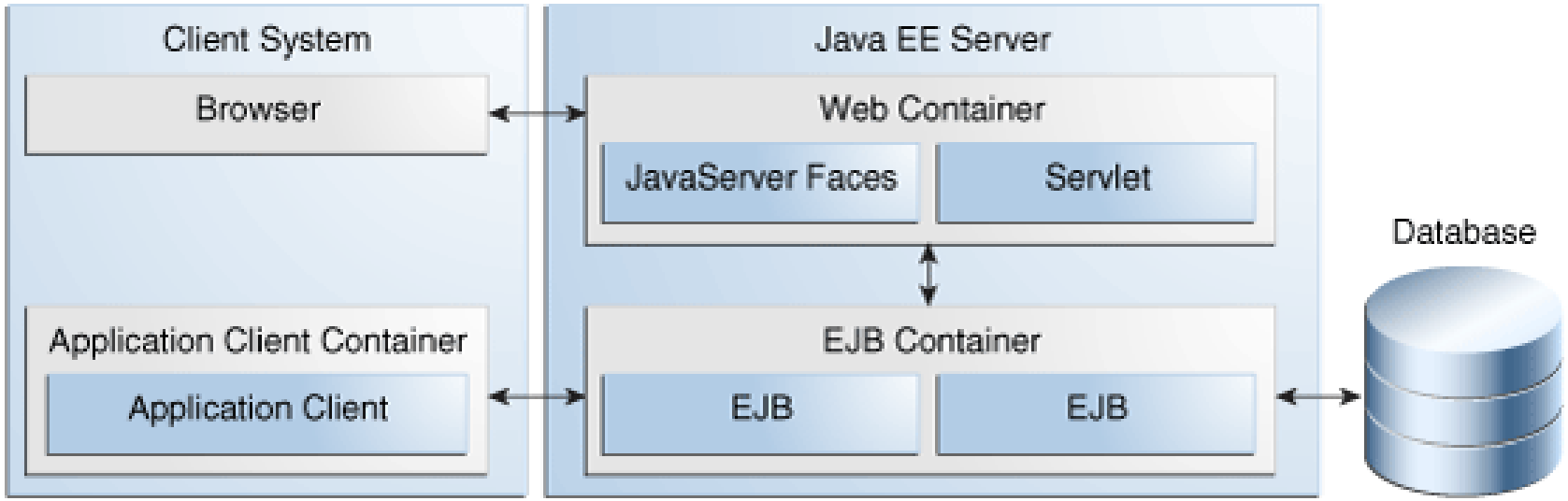
- Implication de « la communauté » pour standards Java
- JCP ? Java Community Process
- Définit les JSR : standards utilisés en Java SE ou Java EE
- Spécifications tiennent compte de nombreux avis d'horizons divers
- Exemples ? JSR 221 : JDBC 4.0 ; JSR 338 : JPA 2.1 ; JSR 345 : EJB 3.2 ; JSR 342 : Java EE 7 ; JSR 346 : CDI...

Standards JCP

- Implication de « la communauté » pour standards Java
- JCP ? Java Community Process
- Définit les JSR : standards utilisés en Java SE ou Java EE
- Spécifications tiennent compte de nombreux avis d'horizons divers
- Exemples ? JSR 221 : JDBC 4.0 ; JSR 338 : JPA 2.1 ; JSR 345 : EJB 3.2 ; JSR 342 : Java EE 7 ; JSR 346 : CDI…
- Tentions entre standard ouvert et contrôle ! (2010, Apache quitte le comité JCP ; Doug Lea également, en faveur de OpenJDK…)

Conteneurs

- Un produit conforme Java EE fournit trois *conteneurs*
  - Conteneur EJB
  - Conteneur web
  - Conteneur application client
- Contenant des *composants* (du type adéquat)
- Chacun fournit des services pour le développeur
- Fournit l'accès aux API (différents conteneurs, différentes API)



Composants

- *Composant* : une unité logicielle assemblée dans une application Java EE avec ses classes et fichiers liés et communiquant avec d'autres composants.
- Code Java compilé normalement
- Assemblé dans une application Java EE : peut utiliser les services ; doit se conformer aux spécifications
- Exécution gérée par le conteneur (pas de `main`, par exemple)

Composant EJB

EJB

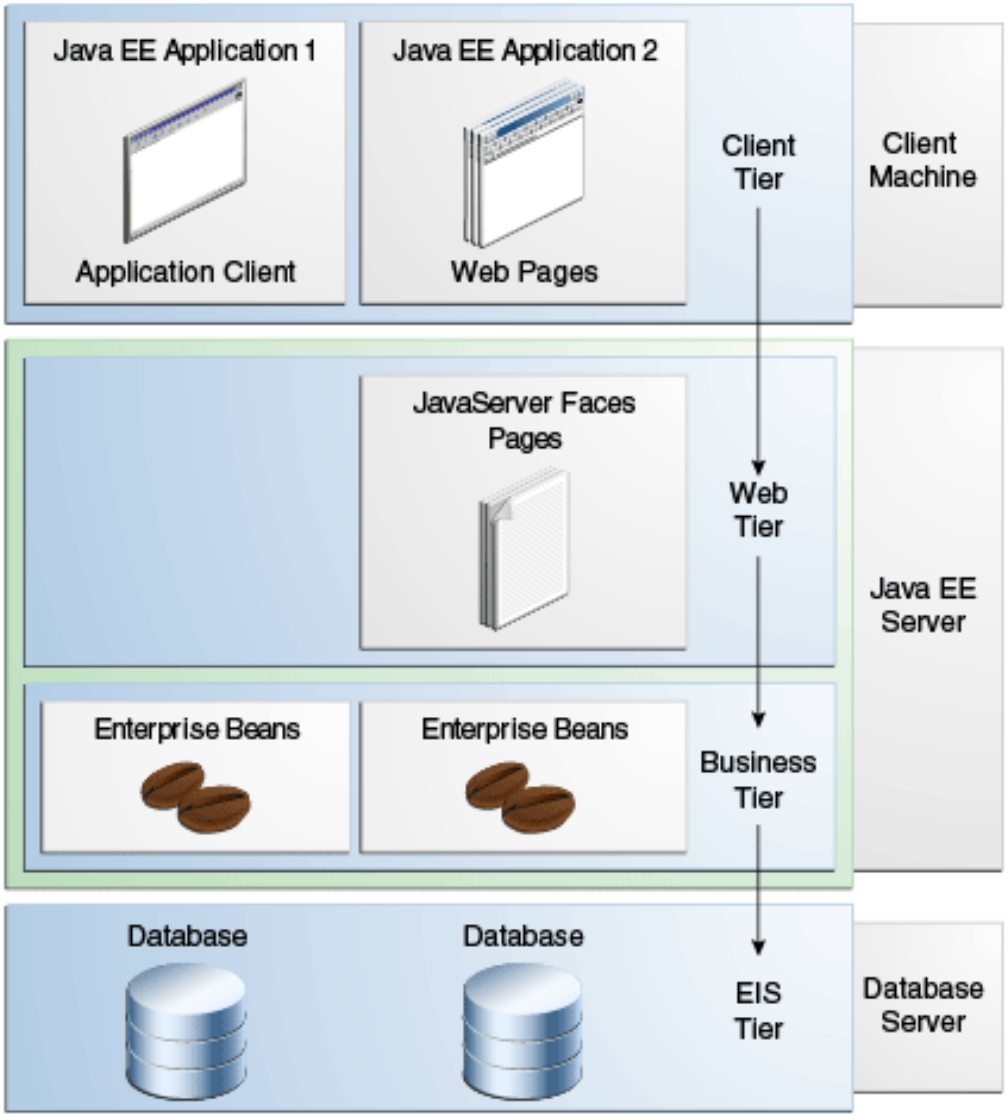
- *Enterprise* Java Bean
- Composant « business », sur le serveur
- Service pouvant être appelé localement ou à distance
- Deux types : session bean, message-driven bean

- Le conteneur rend l'EJB accessible de l'extérieur
- Permet le Remote Method Invocation, sorte de RPC
- Le conteneur instancie, facilite la sérialisation, ...

Composant Web

- Java Servlet n.m.
- JavaServer Faces

Couches (« tier »)



- Ajout d'une couche multithread entre le client et le serveur classique
- Souvent : presentation, logic, data tier
- Couche web : peut être également appelé un client web (pourquoi ?).

Deux applications Java EE

BDD ⇔ EJB ⇔ client

- entreprise A : niveau de stock calculable d'après BDD
- EJB : requête pour obtenir le niveau de stock
- composant client (fournisseur de A) : contacte l'EJB

BDD ⇔ EJB ⇔ Servlet

- entreprise A : niveau de stock calculable d'après BDD
- EJB : requête pour obtenir le niveau de stock
- Web : servlet répondant à HTTP GET (client léger)
- Et puis ? Quel client final ?

Options pour client non-java ?

Modules

- *Module* Java EE : fichier archive compressé
- Ensemble de composants pour un même conteneur (typiquement)
- Éventuellement : un descripteur de déploiement (`.xml`) pour ce type de conteneur (standard Java EE ou par produit)
- Éventuellement : des pages HTML statiques ; des classes utilité, ...
- Les descripteurs surchargent les annotations

Module Web

Module Web

- Fichier `.war`
- Fichiers `.class` servlets et autres dans WEB-INF/lib ou WEB-INF/classes
- Fichiers web statiques (.html, images, ...) dans root
- `WEB-INF/web.xml` : descripteur pour conteneur Web (JNDI)
- `META-INF/glassfish-web.xml` : descripteur pour glassfish
- `META-INF/MANIFEST.MF`

Assemblage et déploiement

- Application Java EE composée d'un ou plusieurs modules
- On peut déployer un module seul (`.war`, `.jar`)
- Ou assembler les modules dans un fichier Enterprise Archive (`.ear`)
- EAR : plusieurs modules et év. descripteur d'application (`META-INF/application.xml`, `META-INF/glassfish-application.xml`)

Déploiement

- Procédure dépend du serveur d'application Java EE
- Typiquement : déplacer l'archive (`.war`, `.ear`, `.jar`) dans un répertoire du serveur
- Accès depuis l'environnement de développement via plug-ins

Services

Exemples :

- Managed beans
- CDI
- RestFul
- JSF
- Bean validation
- JAXB, JAX-WS, JNDI (aussi dans Java SE)

GlassFish Server Tools

- Démarrer, arrêter le serveur
- Déployer des paquets
- Application : console d'administration
- Base de données
- wsimport : artéfacts JAX-WS depuis WSDL (etc.)

À vous

- « Installez » GlassFish (copie depuis `/usr/local/glassfish-4.1/glassfish`)
- Démarrez votre serveur (cf. `bin/`, `http://localhost:8080`, `http://localhost:4848`)
- Désactivez l'écoute extérieure
- Lisez les logs

Accès environnement Java EE via Eclipse

- Accès aux bibliothèques Java EE requis
- En Eclipse : possible d'ajouter un environnement Java EE rudimentaire, « J2EE Preview » (Preferences / Server / Runtime Environments)
- Quand un projet vise ce runtime, eclipse ajoute les bibliothèques en dépendances
- Ces bibliothèques ne seront cependant pas exportées dans l'archive (pourquoi ?)

Notions d'HTTP

HTTP ?

Notions d'HTTP

HTTP ?

- Protocole de communication principal du net
- Échange de requêtes HTTP et réponses HTTP
- Accès à une ressource HTTP via URI
- Requête GET (par exemple)
  - Paramètres possibles (dans « query »)
  - `https://www.google.com/maps?q=paris-dauphine&sourceid=Mozilla-search`
- Réponse ?

Notions d'HTTP

HTTP ?

- Protocole de communication principal du net
- Échange de requêtes HTTP et réponses HTTP
- Accès à une ressource HTTP via URI
- Requête GET (par exemple)
  - Paramètres possibles (dans « query »)
  - `https://www.google.com/maps?q=paris-dauphine&sourceid=Mozilla-search`
- Réponse ?
  - En-tête : media-type (liste), encodage, code de statut…
  - Corps : HTML par exemple
- Cf. RFC 723X

Programmation web « bas » niveau

- Ouverture d'un socket pour écriture réseau
- Définition de l'encodage binaire
- Client envoie requête de connexion
- Serveur écoute sur un socket
- Établissement de la connexion
- Gestion des threads
- La communication peut commencer
- Dispatch à la bonne classe
- Gestion des time-outs
- ...

Servlet

En Java EE, le conteneur effectue une partie du travail pour nous

- Le programmeur (côté serveur) indique ce qu'il faut répondre
- Servlet (ici, HTTP) : classe qui traite les requêtes
- Annoter la classe (`javax.servlet.annotation.WebServlet`) et étendre `HttpServlet` ; préciser `urlPatterns` (ou `value`)
- Requête associée à un servlet par le conteneur
  - `http://server/context-root/servlet-path`
  - context-root : associé à un module web en fonction de son nom d'archive (ou dans descripteur non-standard ou dans descripteur standard d'une application Java EE)
  - servlet-path : associé à un servlet en fonction de `urlPatterns`
- Le conteneur gère le cycle de vie du servlet, lui envoie les objets requête et réponse
- Exemple, `doGet` : récupérer une sortie (`getWriter` ; `getOutputStream`) ; écrire les en-têtes ; écrire le corps

À vous : calculons sur le web

- Créer un module web (dynamique ou statique ?)
- Créer un servlet
- Programmer réponse : "ça fait 0"
- Exporter le module dans une archive déployable (extension ?)
- Déployer le module sur le serveur à la main
- Envoyer une requête GET (comment ?) et voir "ça fait 0"
- Se féliciter

En plus

- Installer Glassfish Tools (depuis Eclipse Marketplace)
- Accepter deux paramètres `add1` et `add2`
- Renvoyer "ça fait " et l'addition des paramètres
- Renvoyer une erreur s'il manque un paramètre

Avant-première

Comment faciliter le développement de servlets ?

Avant-première

Comment faciliter le développement de servlets ?

- Décodage facile de paramètres
- Réponse HTML
- Réponse XML

Avant-première

Comment faciliter le développement de servlets ?

- Décodage facile de paramètres
- Réponse HTML
- Réponse XML

Et plus !

- Injection de références
- Accès à des classes (distantes) pour service (EJB)
- Définition de services web Restful, Soap
- Accès aux données, transactions

Logging

- `System.out.println` ?

Logging

- `System.out.println` ? Le conteneur redirige

Logging

- `System.out.println` ? Le conteneur redirige
- mieux : `LOGGER.warn("Missing parameter here!")`
- En Java EE comme en Java SE : cf. standard logging

Environnements et serveurs dans Eclipse

- Environnement Runtime : donne accès aux bibliothèques J. EE
- Eclipse : *Preferences* / *Server* / *Runtime Environments*
- Puis sélection par projet via *Targeted Runtime*

Serveurs

- Vue *Servers* dans Eclipse
- Liste les serveurs Java EE sur lesquels déployer les modules
- Utiliser : *New* ; *Add and Remove* ; *Publish*
- Sur un serveur : *Properties* puis *Switch location* pour le voir dans un projet spécial (donne plus d'options de configuration)

Servlet et session HTTP

- Un servlet s'exécute dans un contexte
- Entre autres, une session HTTP
- Objet `HttpSession` rendu disponible par conteneur (injection ou accès via paramètres servlet)
- Le serveur tente de tracker la session via un cookie par exemple
- Le serveur fait expirer le cookie après un temps configurable
- Un bon site fonctionne même sans cookies !
- Classe implémente `HttpSessionListener` pour recevoir notifications `sessionCreated`, `sessionDestroyed`. L'annoter `@WebListener`. (Pattern ?

Servlet et session HTTP

- Un servlet s'exécute dans un contexte
- Entre autres, une session HTTP
- Objet `HttpSession` rendu disponible par conteneur (injection ou accès via paramètres servlet)
- Le serveur tente de tracker la session via un cookie par exemple
- Le serveur fait expirer le cookie après un temps configurable
- Un bon site fonctionne même sans cookies !
- Classe implémente `HttpSessionListener` pour recevoir notifications `sessionCreated`, `sessionDestroyed`. L'annoter `@WebListener`. (Pattern ? Observateur !)

Exercices : servlets I

Ces exercices ont pour but de vous familiariser avec l'environnement de développement et vous permettre de vérifier que vous avez compris les concepts de base. Le but n'est atteint que si vous jouez le jeu : évitez de chercher comment faire sur internet ! Utilisez de l'aide seulement si vous êtes bloqués.

- « Installer » GlassFish (copier `/usr/local/glassfish-4.1`)
- Démarrer votre serveur (cf. `bin/`, `http://localhost:8080`, `http://localhost:4848`)
- Désactiver l'écoute extérieure
- Lire les logs du serveur
- Ajouter dans Eclipse un environnement runtime Java EE rudimentaire, « J2EE Preview »

Exercices : servlets II

- Créer un projet web dynamique sans runtime puis le modifier pour qu'il vise ce runtime. Vérifier que Eclipse ajoute les bibliothèques en dépendances (lesquelles ?)
- Créer un servlet
- Vérifier que vous avez accès à la javadoc (par exemple sur `javax.servlet.annotation.WebServlet` et `javax.servlet.http.HttpServlet`).
- Programmer réponse : "ça fait 0"
- Exporter le module dans une archive déployable (extension ?)
- Vérifier si les bibliothèques ajoutées automatiquement par eclipse ont été incluses dans l'archive exportée. Expliquer pourquoi (pas).
- Déployer le module sur le serveur à la main
- Envoyer une requête GET et observer "ça fait 0"

Exercices : servlets III

- Se féliciter [1]
- Ajouter un runtime à eclipse : J2EE Runtime library, référencer les bibliothèques installées dans glassfish. Modifier votre projet existant pour utiliser ce runtime (voir propriétés du projet). Qu'est-ce que ça change ?
- Créer un nouveau serveur de type J2EE Preview dans Eclipse. Expliquer pourquoi il n'y a pas « J2EE Runtime library » dans la liste de serveurs qui peuvent être créés de cette manière. Déployer votre servlet depuis eclipse vers ce nouveau serveur. Vérifier que votre servlet fonctionne depuis votre navigateur, ansi que depuis eclipse (*Run* / *Run*). [2]
- Installer Glassfish Tools depuis Eclipse Marketplace
- Utiliser glassfish comme runtime. Voir ce que ça change. Déployer vers glassfish et vérifier que le servlet fonctionne toujours

Exercices : servlets IV

- Envoyer une information de log indiquant le résultat du calcul avant son renvoi à l'expéditeur
- Renvoyer une réponse HTML bien formée : <html><body> Ça fait... </body></html>. Comment s'assurer que le client web va bien interpréter de l'HTML ?
- Renvoyer ce même contenu mais en texte pur, de façon à ce que ces balises s'affichent sur le client web. Comment s'assurer de l'affichage correct des caractères non ascii, Ç par exemple ?

*[3] Est-il possible de déployer votre module web comme une archive `.ear` ? Si oui, essayer de le faire, si non, expliquer pourquoi.

- Accepter deux paramètres dans le servlet : `add1` et `add2`. Renvoyer "ça fait " et l'addition des paramètres
- Renvoyer une erreur s'il manque un paramètre (indice : voir la javadoc de `HttpServletResponse`).

Exercices : servlets V

* Comment être sûr du type d'erreur à renvoyer ?
- Ajouter au module une page HTML statique `index.html` qui dit « Hello ! » lorsqu'on la visite. Où sera-t-elle placée dans l'archive ?
- (Commencer à) développer un servlet qui a sa place dans votre projet

---

1. Cette étape importante est à répéter à l'issue de chaque exercice qui vous a posé une difficulté.

2. Il semble qu'un bug dans J2EE Preview empêche parfois l'accès à votre servlet via ce serveur. Vérifiez simplement dans ce cas que vous pouvez accéder à `http://server/context-root/`.

3. Les astérisques indiquent des exercices plus difficiles, à essayer de faire pour gagner des points de prestige.

Servlet (trop de détails)

- Le servlet peut accéder à des objets via getAttribute et setAttribute d'une classe représentant un scope : sur ServletContext, HttpSession, ServletRequest. Il faut se protéger contre les accès concurrents !
- HTTP request URL : http://host:portrequestpath?querystring, with
- requestpath=contextpathservletpathpathinfo, with
- contextpath=/contextroot (of the servlet's web app) or "",
- servletpath=The path section that corresponds to the component alias that activated this request. This path starts with a forward slash (/) (or is empty), pathinfo = the rest.

Licence