Implementation de services Web en JEE

E.Coquery

emmanuel.coquery@univ-lyon1.fr

http://liris.cnrs.fr/~ecoquery
→ Enseignement → TIW5

Serveur d'application

- Conteneur : intègre des mécanismes de :
  - chargement de classes
  - d'isolation / d'imbrication entre contextes de classes
  - peut être implémenté via *e.g.* OSGi
- Orienté composants
  - Chaque composant a un rôle précis
    - Permet de simplifier chacun des composants
  - De nombreux composants sont fournis de manière standard
    - Soit directement
    - Soit générés à partir de spécifications de haut niveau
- APIs destinées à simplifier la programmation des différents composants de l'application
- Courbe d'apprentissage importante

Servlets - rappels

Point d'entrée d'un client sur un serveur web J2EE

Principe

- Une classe Java
  - Qui implémente l'interface `javax.servlet.Servlet`
  - En général en étendant la classe `javax.servlet.http.HttpServlet`
- Un morceau dans un fichier de configuration (XML) du conteneur / une annotation en JEE 6
  - Relie la classe à un emplacement (i.e. une URL) du serveur

Servlet - rappels - 2

Méthodes Java appelées lors d'une requête d'un client :

- doGet, doPost, doPut, doDelete
  - En fonction de la méthode HTTP utilisée
  - Arguments représentant la requête et la réponse
- Init et destroy
  - Réservation / libération de ressources (*e.g.* connection JDBC)

Attention à l'aspect concurrent :

- Gestion de plusieurs requêtes simultanées

Servlets : suffisant pour des services Web basiques

Un point d'accès à un service : Reçoit une requête HTTP

- Extrait de la requête un document SOAP
- Effectue un traitement
- Créé un document SOAP résultat
- Renvoie ce document comme réponse

Réalisable par une servlet en utilisant

- les APIs XML Java
- SAAJ

Contrôleur

1 service $\leftrightarrow$ n opérations

- n servlets + 1 servlet contrôleur en amont
  - ou routage via l'url ($\Leftrightarrow$ n services)
- contrôleur dans le `doXXX`
  - 1 opération $\leftrightarrow$ 1 méthode

@WebServiceProvider

Annotation JAX-WS sur une classe C

- C implémente `javax.xml.ws.Provider<T>`
  - `T invoke(T request)`
- `@ServiceMode(PAYLOAD)`
  - T instance de `javax.xml.transform.Source`
- `@ServiceMode(MESSAGE)`
  - T instance de `javax.xml.soap.SOAPMessage`
- Servlet contrôleur fournie par le framework (e.g. CXF)

→ Evite d'avoir à créer le message SOAP à partir de la requête HTTP

→ Intégré dans les frameworks d'injection de dépendance

Chaîne de traitement

- Réception de la requête HTTP (serveur web)
- Analyse du header SOAP + redirection sur le bon composant (contrôleur)
- Désérialisation XML $\rightarrow$ objets
- Traitement métier
- Sérialisation de la réponse objet $\rightarrow$ XML
- Réponse HTTP

@WebService sur une classe

Annotation JAX-WS sur un objet métier (typiquement un Bean)

- méthodes métier $\leftrightarrow$ opérations
- servlet contrôleur fournie (e.g. `org.apache.cxf.transport.servlet.CXFServlet`)
  - requête HTTP
  - (dé)sérialisation XML $\leftrightarrow$ objet
  - appel méthode
- Utilisation dans un framework d'inversion de contrôle
  - `@Resource WebServiceContext context`

@WebService sur une interface

$\leftrightarrow$ portType/interface WSDL

- Spécification d'un WSDL
- et/ou annotations de configuration :
  - @WebMethod
  - @WebParam
    - Possibilité de récupérer des valeurs dans le header SOAP
  - @WebResult

  annotation utilisables dans les classes métier
- @WebService(endpointInterface="...") sur la classe métier

Déploiement : mapping URL $\leftrightarrow$ service dans un fichier de configuration XML dépendant du framework

Génération de schéma

Remarque

Structure des classes
+ Annotations JAXB
→ Structure XML

Possibilité de générer

- le schéma des messages
- le WSDL

via les framework JAX-WS

Génération de code

- Génération de classes annotées pour représenter les documents spécifiés par le schéma
- Génération d'une interface annotée avec @WebService.
  - Il reste à l'implémenter
- Génération de stub/proxy pour le côté client

CXF : WSDL2Java ← utilisable depuis maven

CXF

- Framework d'implementation des services web
- implémente JAX-WS (SOAP), JAX-RS (REST)
- Génération à la volée de clients via une interface
  - classe `JaxWsProxyFactoryBean`
  - spécification de la classe/interface du service
- Utilise Spring

Références

- https://jax-ws.java.net/
- http://cxf.apache.org/docs/index.html
- http://docs.oracle.com/javaee/6/tutorial/doc/bnayk.html