Bonjour

- Architecture
- En détail: OSGi™
- En détail: Java EE
- Questions

Projet M2M:

Système de capture et de remontée de mesures

OSGi™ + JOnAS

Lionel DEBROUX

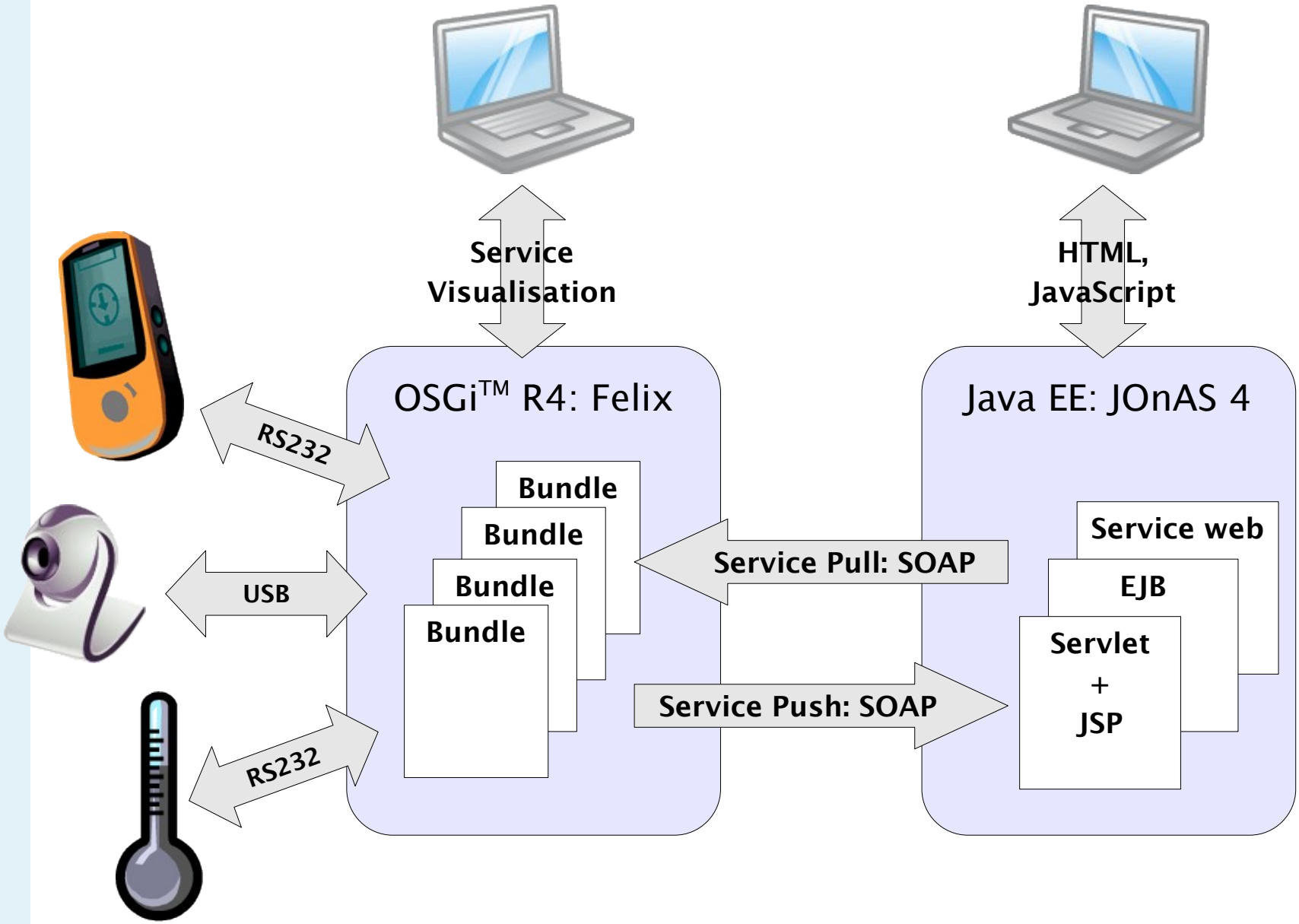
Savaş Ali TOKMEN

M2P Génie Informatique

UFR IMA, Grenoble, FRANCE

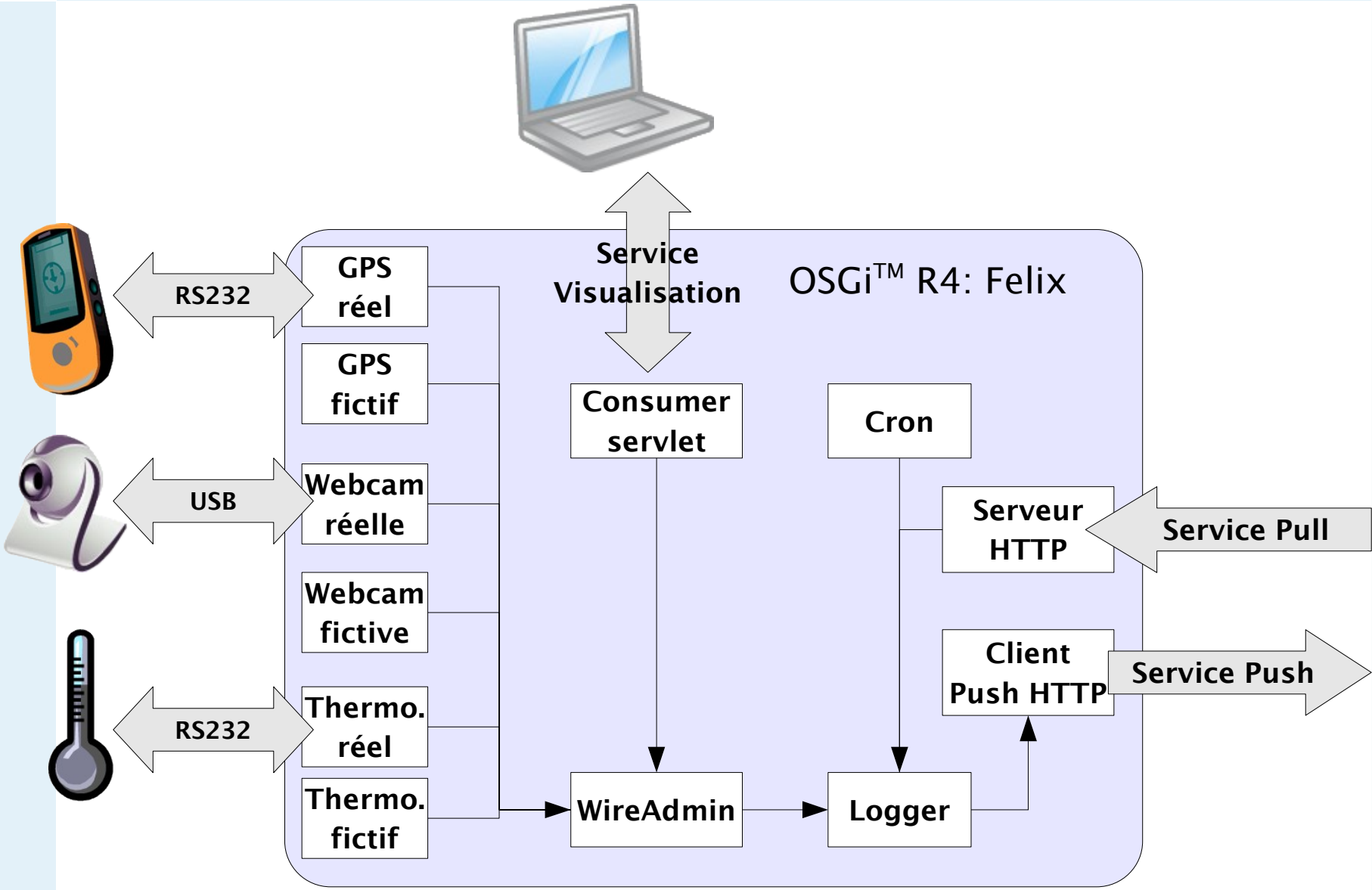
Architecture globale

- **Architecture**
  - **Globale**
  - OSGi™
  - Java EE
- En détail: OSGi™
- En détail: Java EE
- Questions



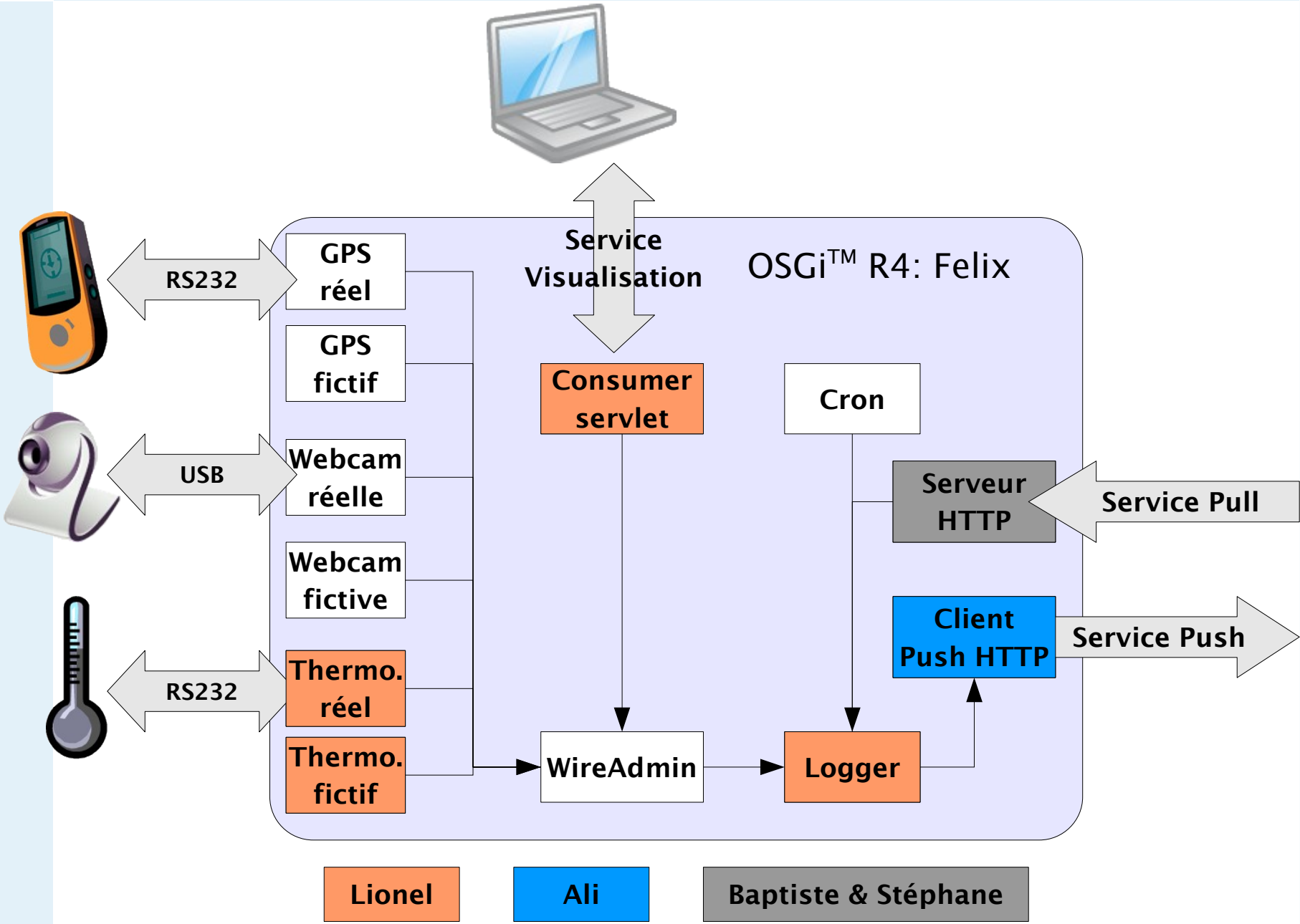
Architecture interne: OSGi™

- **Architecture**
  - Globale
  - **OSGi™**
  - Java EE
- En détail: OSGi™
- En détail: Java EE
- Questions



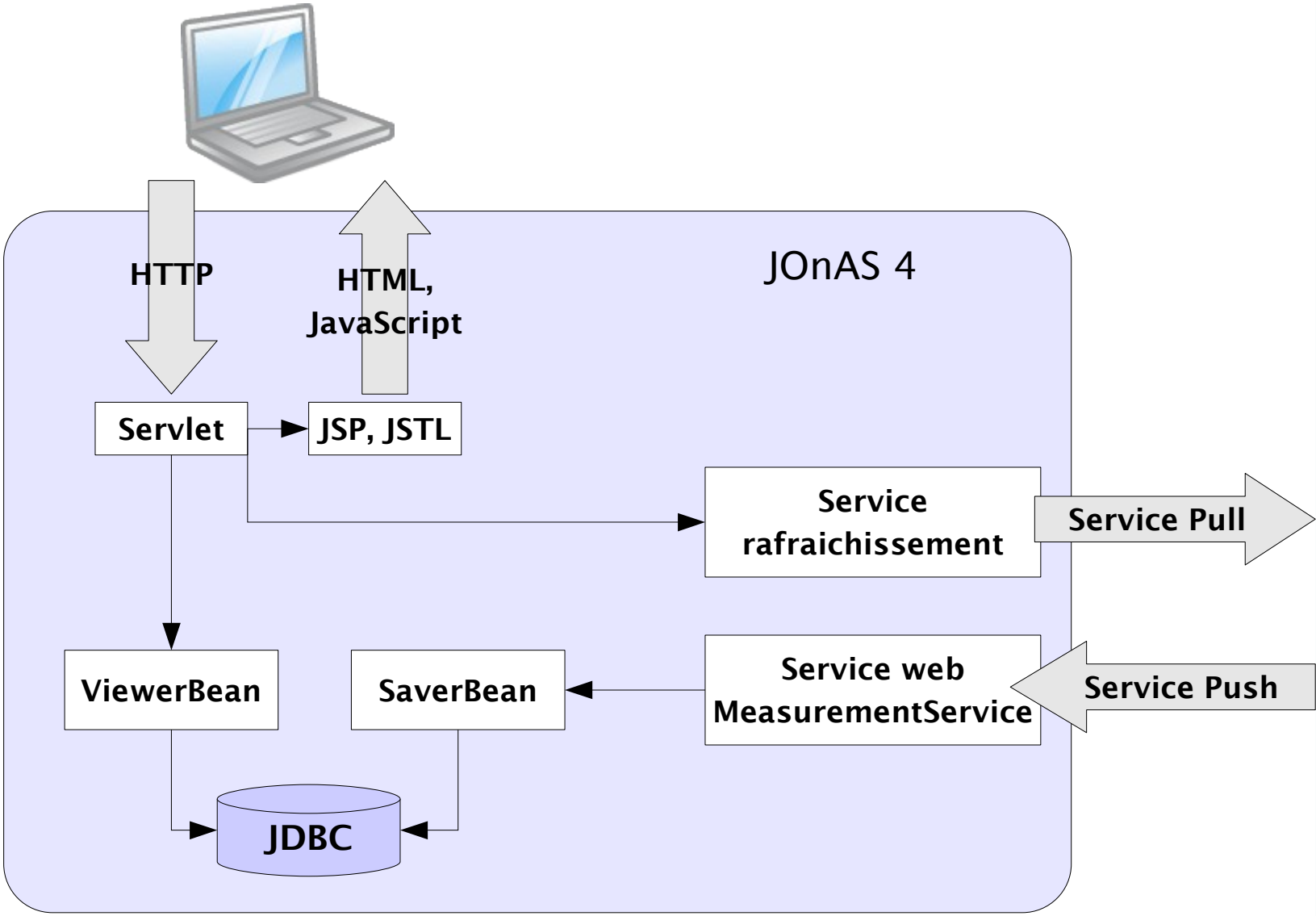
Répartition du travail: OSGi™

- **Architecture**
  - Globale
  - **OSGi™**
  - Java EE
- En détail: OSGi™
- En détail: Java EE
- Questions



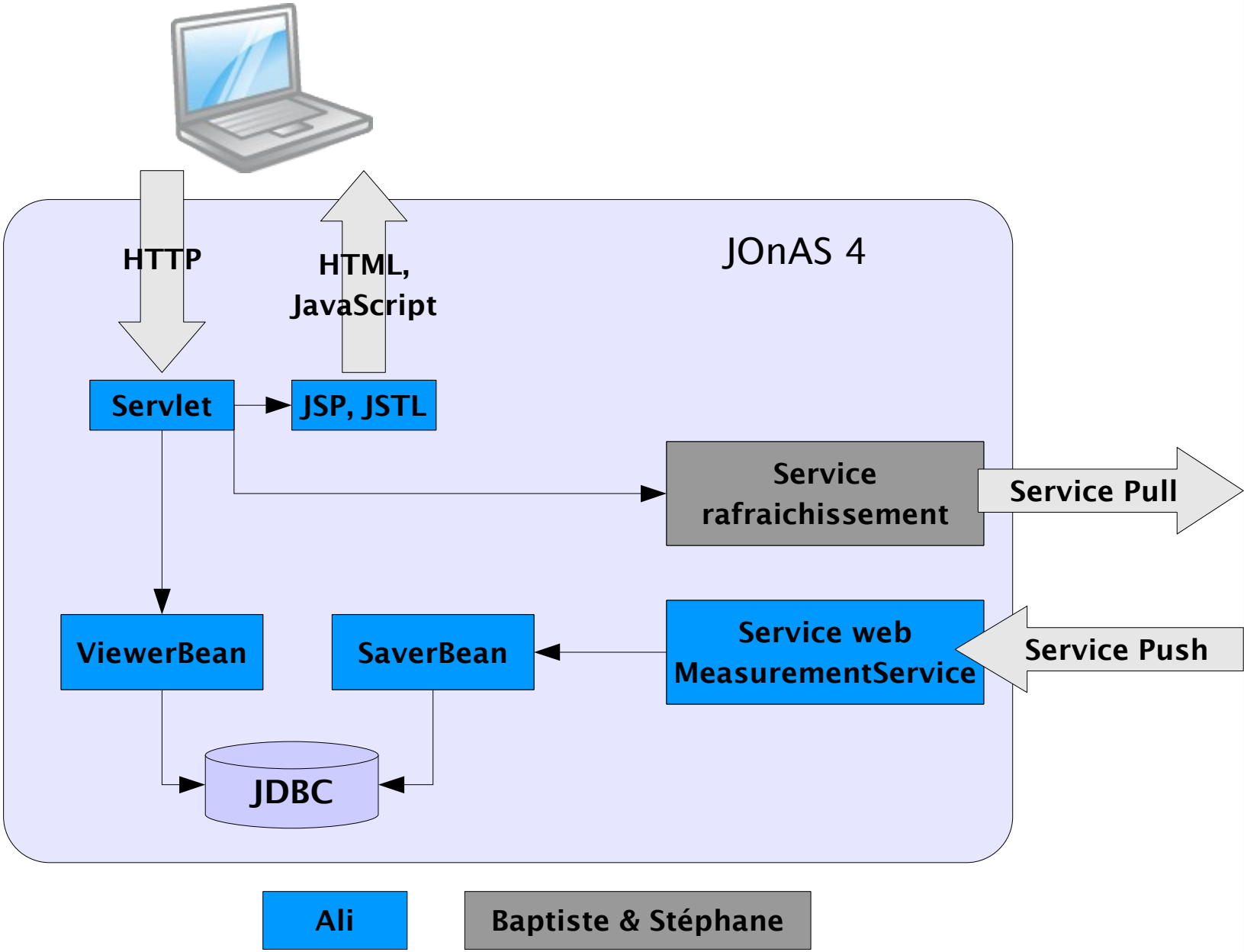
Architecture interne: Java EE

- **Architecture**
  - Globale
  - OSGi™
  - **Java EE**
- En détail: OSGi™
- En détail: Java EE
- Questions



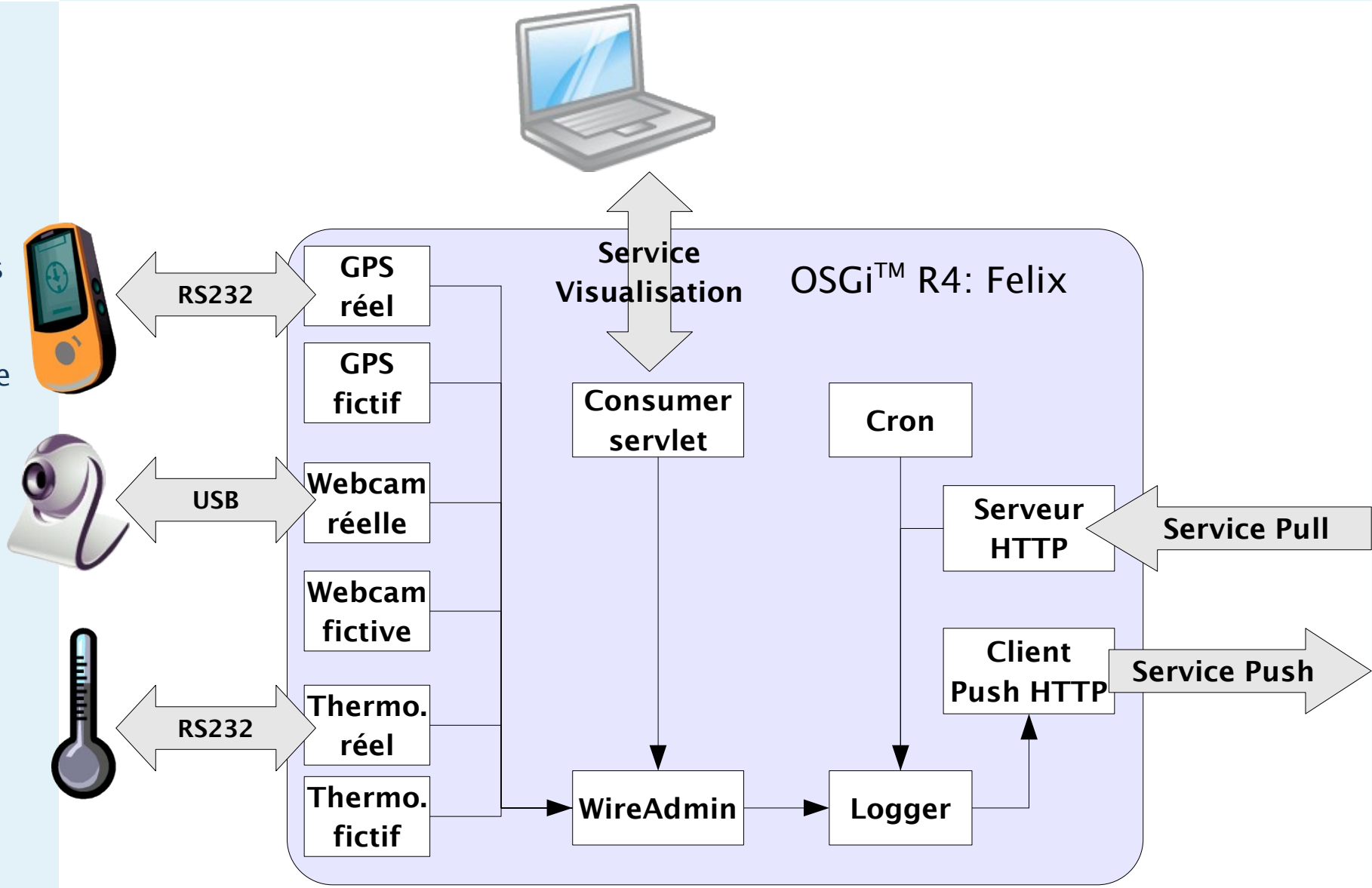
Répartition du travail: Java EE

- **Architecture**
  - Globale
  - OSGi™
  - **Java EE**
- En détail: OSGi™
- En détail: Java EE
- Questions



Rappel: architecture OSGi

- Architecture
- **En détail: OSGi™**
  - Producers
  - Consumers
  - Consumer Servlet
  - PushService
  - PullService
- En détail: Java EE
- Questions



Producers

- Implémentent **`org.osgi.service.wireadmin.Producer`**
- Utilisent le WireAdmin pour communiquer avec les Consumers
- Utilisent le CronService pour pousser les infos vers les Consumers (loggers)
- Produisent
  - **`Measurement`** (température);
  - **`Position`**;
  - images (base64).

Consommateurs (loggers)

- Implémentent **`org.osgi.service.wireadmin.Consumer`**
- Utilisent le WireAdmin pour communiquer avec les Producers
  - Créent les Wire (objet de communication)
- Utilisent le CronService pour demander les infos aux Producers
- Sont appelables par le Pull Service
  - méthode **`sendData`**
- Appellent le Push Service

Consumer Servlet

- Consumer spécial qui permet de visualiser les productions des Producers
  - création manuelle des Wire
- Utilisent un HTTP Service pour fournir les fichiers et exécuter la Servlet

Push Service

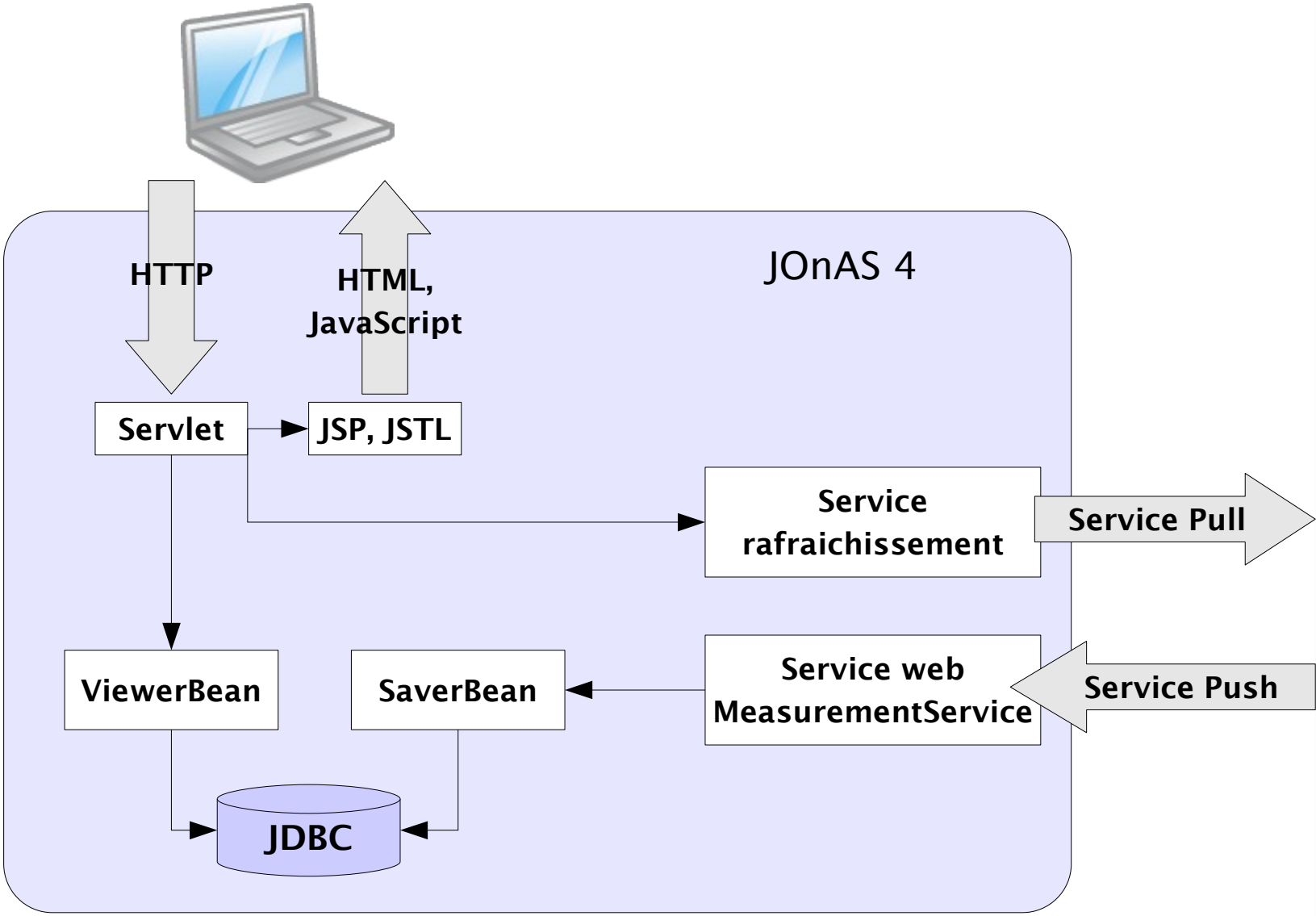
- Appelé par les loggers pour pousser les informations vers le serveur JOnAS
  - interface **`PushService`**
  - méthode **`pushMeasurement`**
- Code de génération de SOAP partiellement auto-généré (AXIS).

Pull Service

- Appelé par le serveur JOnAS pour que les loggers fassent remonter les dernières mesures dont ils disposent
  - interface **`LoggerPullService`**
  - méthode **`sendData`**
- Utilise XFire sur Jetty 5.x

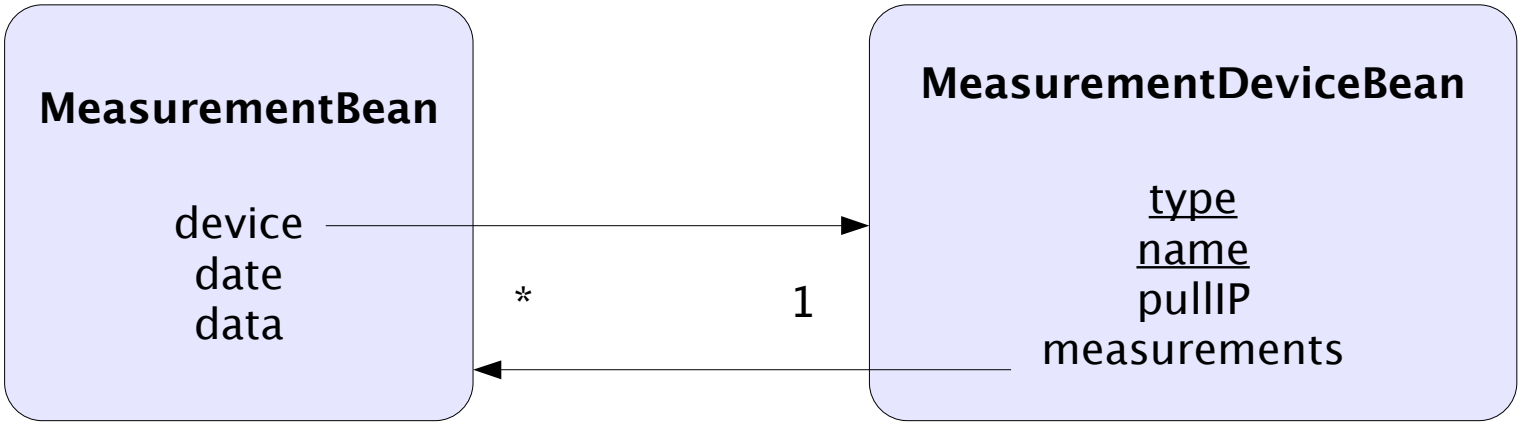
Rappel: architecture Java EE

- Architecture
- En détail: OSGi™
- **En détail: Java EE**
  - EJB Entité
  - EJB Stateless
  - Service web
  - Servlet, JSP
  - Installation
- Questions



EJBs entité

- EJB3
- Gérés par le container
- Même schéma que les structures du service web + relations entre structures



EJBs stateless

- EJB3
- Tous les deux transactionnels:

  `javax.ejb.TransactionAttributeType.REQUIRED`
- Deux EJBs:
  - EJB d'écriture: **MeasurementListener**
  - EJB de lecture: **MeasurementViewer**
- Chacun des EJBs accédé via JNDI:
  - Possibilité de client lourd
  - Possibilité de mettre sur des serveurs séparés

Service web: interface extérieure

- Namespace WSDL `MeasurementService`
- Deux structures:
  - Measurement:
    - Date: java.util.Date
    - Data: java.lang.String
  - MesurementDevice:
    - Type: java.lang.String
    - Name: java.lang.String
    - PullIPAddress: java.lang.String
- Une méthode:

```
pushMeasurement(MeasurementDevice, Measurement[])
```

- Service web standard (namespace simple, arguments explicites), généré par java2wsdl

Service web: implémentation

- JAX RPC
- Sauvegarde les données en utilisant l'EJB MeasurementListenerBean:
  - Transactionnel
  - Possibilité de séparer les machines d'écoute du service web et de sauvegarde des données
- Servlet auto-implémenté par JOnAS
  - Via web.xml, webservices.xml et mapping.xml
  - Aucun servlet n'a été écrit :)

Interface web: servlet, JSP et JSTL

- Appels reçus sur servlet
- Servlet accès aux données en utilisant l'EJB MeasurementViewerBean:
  - Transactionnel
  - Possibilité de séparer les machines d'écoute du servlet et de lecture des données
- Selon le type d'appareil, redirection vers JSP de rendu adéquat
  - GPS, thermomètre ou webcam
  - Autres appareils (liste simple)
  - Chaque équipe écrit le sien :)
- Possibilité d'utiliser JSTL dans JSP

Installation et deploiement

- Installation des EJBs et du WAR par tâche ANT
- EJBs auto-deployés par JOnAS
- WAR à deployer manuellement, possibilité d'automatisation ou de le mettre dans les WAR « autodeploy »
- JSPs compilés à la première execution

Questions ?

Questions ?

Merci

- Architecture
- En détail: OSGi™
- En détail: Java EE
- Questions

Merci

Documents disponibles sur

http://scholar.alishomepage.com/Master/JOnAS_OSGi/