


 Université Lille 1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Document présentation

Description	Examen de CARA
--------------------	----------------

Document certification

	<i>Name</i>	<i>Fonction</i>
<i>Author</i>	DJEBIEN Tarik	Etudiant Miage IPI NT
<i>Decidor</i>	ROOS Jean-François	Enseignant CARA

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

[Synthese](#)

[Composants](#)

[Services](#)

[Technique](#)

[Problème](#)

[Définition de l'architecture](#)

[Choix d'une technologie](#)

[Cours](#)

[Webservices](#)


[Middleware](#)

[Technologie du middleware](#)

[Composants](#)

[CORBA-like \(RMI ou CORBA\)](#)


[EJB](#)

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Synthese

Composants

	Apports des environnements a composants (EJB, webservices ...) par rapports aux environnements a objets repartis
CONCEPTION	-facile a concevoir, comprendre et réutiliser : composants plus facilement réutilisable par des clients légers/lourds (monter en abstraction / objet via des design patterns : application service , session facade) -séparation des préoccupations non fonctionnelles (design pattern : Single Responsibility Principle) -on se focalise sur la logique applicative, la logique de présentation est du ressort du client
REALISATION	-Service fourni par le serveur au container (persistance, transaction, nommage, sécurité, communication bas niveau) -configuration : configurer (XML annotations) plutôt que programmer (Inversion du contrôle) -gestion des interactions et des dépendances : faire gérer l'architecture applicative par le container (Injection de dépendances)

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

EXECUTION	-dynamicité (plus rapide que RMI) -empaquetage (packaging : jar, war, ear) -assemblage (modularité) -déploiement (glassfish, jboss, weblogic)
-----------	---


Services

Atouts des environnements orientés services


Modèle «économique» tout est service

Principes des SOA :

- **Couplage** lâche de façon à minimiser les interdépendances, dépenses
- **Contrat** explicite entre les services qui interagissent, communication ne se font que via un accord entre services
- **Autonomie** ie le service contrôle complètement la logique qu'il encapsule
- **Abstraction** : au-delà de ce qui est visible dans le contrat, le reste est caché, on masque la logique du service au monde extérieur
- **Réutilisation** : la logique encapsulée est elle-même organisée en services de façon à promouvoir la réutilisation
- **Composabilité** : la capacité à regrouper plusieurs services et les coordonner de façon à former un nouveau service 'composite '
- **Absence de contexte** : interdiction de conserver des informations spécifiques à une activité.
- **Auto-description** : organisation du service de façon à ce que ses capacités puissent être découvertes avec les outils de recherche disponibles
- **Optimization** : services peuvent être optimisés individuellement

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

- **Encapsulation existant** encapsulé pour pouvoir être réutilisé avec SOA

 Université Lille 1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0


Avantages et inconvénients d'un environnement à composants type EJB aux Web Services

Avantages

- Les EJB, grâce à leur containers, fournissent des services de gestion des transactions (JTA), persistance (JPA), de gestion des pools de connexion, des datasources, de l'annuaire de nommage des ressources (JNDI), des messages orientés middleware (JMS) et de sécurité (JASS) alors que les web services ne gèrent pas la sécurité (sécurité basée sur la sécurité du protocole sous-jacent (ex. HTTPS)) , Transaction.

Inconvénients

- Les EJBs sont construit au-dessus de RMI, RMI et EJB sont dépendants du langage JAVA :
 Si vos clients doivent être écrits dans quelque chose d'autre (par exemple.NET, PHP, etc) mieux vaut utiliser les services Web (que les composants de type EJB), qui utilisent un protocole standard comme le HTTP ou XML basé sur SOAP.
 Les EJBs n'ont pas d'interopérabilité, comparé au web services.
 - Les EJBS nécessitent la configuration du réseau (numero de port ...)

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Technique

EJB Entite Notification.java

@Entity

public class Notification implements Serializable {

 @Id

 @GeneratedValue(strategy=GenerationType.AUTO)

 @Column(name="NOTIFICATION_ID")

 private Long id;

 @Temporal(TemporalType.DATE)

 private Date date;

 private String description ;

 private String detail ;

 @ManyToMany(mappedBy="notifications")


 protected Set<Utilisateur >utilisateurs ;

 ...

Constructeurs, getters, setters, equals, hashCode

 ...

}

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

EJB Entite Utilisateur .java

@Entity

public class Utilisateur implements Serializable {

 @Id

 @GeneratedValue(strategy=GenerationType.AUTO)

 @Column(name="UTILISATEUR_ID")

 private Long id;

 private String nom ;

 private String password ;

 @ManyToMany

 @JoinTable(name="UTILISATEURS_NOTIFICATIONS",

 joinColumns=

 @JoinColumn(

 name="UN_UTILISATEUR_ID",

 referencedColumnName ="UTILISATEUR_ID"),

 inverseJoinColumns=

 @JoinColumn(

 name="UN_NOTIFICATION_ID",

 referencedColumnName ="NOTIFICATION_ID"))


 protected Set<Notification > notifications;

 ...

 Constructeurs, getters, setters, equals, hashCode

 ...

}

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

@Remote

public interface PushNotificationServiceRemote {

/**

* Indique que la notification a bien ete lue par l'utilisateur

*/

void push(Utilisateur usage, List<Notification> notificationsLues) ;


/**

* Connaitre le detail explicite de la notification

*/

String getNotificationDetail(Notification notificationChoisie) ;

}

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

```


@Stateful(mappedName="PushNotificationService")
public class PushNotificationServiceImpl implements PushNotificationServiceRemote {

    @PersistenceContext(unitName = "Cara-examPU")
    protected EntityManager entityManager;

    @Override
    public void push(Utilisateur usager, List<Notification> notificationsLues) {
        usager.getNotifications().removeAll(notificationsLues);
        entityManager.merge(usager);
    }

    @Override
    public String getNotificationDetail(Notification notificationChoisie) {
        return notificationChoisie.getDetail();
    }
}

```

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Producteur de notification cote serveur

- Contient : Producteur de notification via JMS, un EJB timer service de déclenchements de notification périodiques.

@Singleton

@Startup

@LocalBean

public class Producer {


 @Resource(mappedName="jms/TopicConnectionFactory") //l'id de la factory
 private TopicConnectionFactory connectionFactory;

 @Resource(mappedName = "jms/Topic")// l'id du topic
 private Topic destination;

 @Resource
 private TimerService ts;

 @PostConstruct
 public void configureTimer(Notification n) {
 ts.createTimer(1000,25000,n);
 }


 @Timeout
 public void monitorerNotification(Timer timer) {
 sendNotificationToUsers();
 }

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

```

public void sendNotificationToUsers(){
try {
    Connection connection = connectionFactory.createConnection();
    Session session = connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
    MessageProducer producer = session.createProducer(destination);
    TextMessage message = session.createTextMessage();
    message.setText("This is a notification");
    producer.send(message);
}
    /* Send a non-text control message indicating end of messages. */
    producer.send(session.createMessage());
} catch (JMSEException e) {
    System.err.println("Exception occurred: " + e.toString());
} finally {
    if (connection != null) {
        try {
            if (session != null) {
                session.close();
            }
            connection.close();
        } catch (JMSEException e) {}
    }
}
}

```

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

On suppose que le client s'exécute dans le même serveur d'application.
(sinon faire un lookup dans l'annuaire JNDI)

Client consommateur


```
@MessageDriven(mappedName="jms/Topic")
public class NotificationConsumer implements MessageListener {

    private List<Notification> messages ;

    @Resource(mappedName="jms/TopicConnectionFactory") //reference a la factory
    private TopicConnectionFactory connectionFactory;

    @Resource(mappedName = "jms/Topic")
    private Topic dest ;

    /**
     * Casts the message to a TextMessage and displays its text.
     * @param message the incoming message
     */
    public void onMessage(Message message) {
        TextMessage msg = null;
        try {
            if (message instanceof TextMessage) {
                msg = (TextMessage) message;
                Notification n = new Notification() ;
                n.setDescription (msg.getText());
                messages.add(n);
            } else {
                System.err.println("Message is not a TextMessage");
            }
        } catch (JMSEException e) {
            System.err.println("JMSEException in onMessage(): " + e.toString());
        } catch (Throwable t) {
```

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0


```

        System.err.println("Exception in onMessage()." + t.getMessage());
    }
}

public static void main(String[] args) {
    messages = new ArrayList<Notification>();
    String destType = null;
    Connection connection = null;
    Session session = null;
    MessageConsumer consumer = null;
    TextMessage message = null;

    try {
        connection = connectionFactory.createConnection();
        session = connection.createSession(false,
            Session.AUTO_ACKNOWLEDGE);
        consumer = session.createConsumer(dest);
        consumer.setMessageListener(this);
        connection.start();
    } catch (JMSEException e) {
        System.err.println("Exception occurred: " + e.toString());
    } finally {
        if (connection != null) {
            try { connection.close(); } catch (JMSEException e) { }
        }
    }
}


```

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Problème

Définition de l'architecture

Choix d'une technologie

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Cours

Webservices

Avantages


Communiquer dans un environnement réparti & hétérogène :

	Protocole	Format représentation données
DCOM	RPC DCE	binaire
CORBA	IIOP	binaire (CDR)
RMI	JRMP/IIOP	binaire (Java Serialization)

- Protocoles + format de représentation données spécifiques
Interopérabilité ok, mais...
- Solution liée à l'environnement
- Nécessite un protocole de transport (TCP ?)
- Configuration du réseau (no de port?)
- Formats de données universel (SOAP)
 - HTTP: simple, sans état, toutes les chances de traverser les firewalls
 - XML : ASCII (pas binaire), parsing facile, standard W3C
- Des standards simples (SOAP, WSDL, UDDI)
- Multi Protocole / Multi OS / Multi Langage
- Paradigme de Service
- Des outils (éditeurs et moteurs)

Inconvénients

- Typage (pas de consensus)

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

- Performance
- Jeunesse (Sécurité, Transaction,...)


Avantages de SOAP (Simple Object Access Protocol)

- Indépendant des langages
- Indépendant des OS
- Indépendant des protocoles de communication
(mais HTTP protocole simple, facilement implantable)
- sécurité basée sur la sécurité du protocole sous-jacent (ex. HTTPS)

Inconvénients de SOAP

SOAP n'a aucune notion orientée objet

- pas de référence d'objet
- pas d'instanciation (pas d'activation à la demande)
- pas de cycle de vie (pas de gestion de l'exécution des requêtes)
- pas de GC (pas de ramasse-miettes)
- pas de variables d'instances
- pas d'état "conversationnel" i.e. session (+/- EJB stateless bean)
(pas de transaction entre +ieurs invocations)
- fiabilité essentiellement basée sur TCP

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0


Middleware

•Middleware : désigne dans le cadre de l'informatique répartie, toutes les couches logicielles qui permettent de communiquer à distance.

Avantages

Ensemble de concepts génériques

- adressage définir une référence unique
- transport échange de données
- liaison associer une entité locale à une référence
- représentation transformation des données dans un format commun
- protocole interactions entre entités
- activation à la réception d'un message, définition de l'entité traitant
- exécution associe les ressources nécessaires
- objet, composant, service
- interface, contrat
- souche, squelette
- service techniques (aussi appelés non fonctionnels ou extra-fonctionnels)
- annuaires (registre, moteur de recherche, pages jaunes, ...)
- sécurité (contrôle d'accès, cryptage, authentification, ...)
- transaction
- persistance des données
- concurrence
- synchronisation
- réplication
- migration
- ...

 Université Lille 1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Inconvénients

Passage à large échelle : Web

Protocoles propriétaires

- IIOP, RMI, DCOM
- Firewall

Pas d'ouverture des services


- Notion de moteur de recherche inexistante

Trop de contraintes sur le client !

- Doit posséder les souches
- Difficulté de construire dynamiquement
- Complexité
- CORBA : IDL, Mapping, ...
- EJB : Container, JNDI, ...
- Pérennité : remise en question
- CORBA, EJB, .Net, ...
- Prix
- Plates-formes
- Compétences

Portage d'applications existantes difficile

Solutions non standards


 Université Lille 1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

MOM (Message Oriented Middleware)

- Les Middleware Orientés Messages sont des systèmes fournissant à leurs clients un service de Peer to Peer : c'est un logiciel serveur dont le rôle est de fédérer l'envoi et la réception de ces messages entre les différents types d'applications.
- La communication de deux applications via un MOM est complètement asynchrone, c'est à dire que l'émetteur et le destinataire n'ont pas besoin d'être connectés simultanément lorsqu'ils communiquent.
- La communication n'est synchrone qu'entre l'émetteur et le MOM d'une part, et le MOM et le destinataire d'autre part.

Définitions

- **MOM store and forward** : le MOM stocke le message et le route par la suite à son destinataire lorsque celui-ci le demande.
- **Provider** : produit, logiciel, qui fournit des services d'envoi/réception de messages en mode asynchrone.
- **Message** : informations échangées, trois parties distinctes :
 - les données elles mêmes
 - l'entête du message (son identifiant, date de dépôt ...)
 - ses propriétés, les caractéristiques fonctionnelles du message, qui sont différentes pour chaque application émettrice
- **Queue** : espace de stockage pour les messages, limité ou non.
- **Topic** : file d'attente particulière destinée à recevoir les messages utilisés dans le mode Publish/Subscribe
- **Queue Manager** : ensemble de files d'attentes formant un tout cohérent et géré par une ou plusieurs instances d'un MOM. Le queue manager est l'équivalent dans le monde des MOM d'une base de données dans le monde des SGBD.
- **Channel** : lien entre deux queue manager qui communiquent et s'envoient des messages.
- **Noeud** : lors de la communication entre deux gestionnaires de files d'attente par le biais d'un canal, on parle de communication entre deux noeuds.

 Université Lille 1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Technologie du middleware

Ensemble de technologies middleware pour la construction d'applications réparties

communications distantes :


- RMI-IIOP : requête/réponse (TCP + IIOP + sérialisation Java)
- JAX-WS : requête/réponse Web Service (HTTP + SOAP + XML)
- JMS : MOM (message oriented middleware) : message + boîte à lettres

services systèmes :

- JNDI (Java Naming and Directory Interface): annuaire
- JTA (Java Transaction API) : gestion de transactions
- JPA (Java Persistence API) : gestion de la persistance des données

un système de connecteurs (JCA) pour permettre à la plate-forme d'interagir

- JDBC (Java Data Base Connectivity) : accès client/serveur aux SGBD
- JavaMail : envoi/réception de mail support des web services

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

Composants

Les applications JEE sont constituées de composants JEE : unité logicielle indépendante contenant classes et fichiers :

- applications clientes, applets : composants qui tournent sur le client.
- composants Web : tournent sur le serveur JavaServlet, JavaServer Faces, JavaServer Pages
- composants Enterprise Java Beans : composants métiers situés sur le serveur

Les composants JEE suivant les spécifications JEE, sont assemblés en application, déployés sur et gérés par le serveur JEE.

Avantages

- monter en abstraction / objet
- architecture logicielle

séparation des préoccupations (SRP (Single Responsibility Principle))

- développer le métier indépendamment des préoccupations non fonctionnelles
- composants plus facilement réutilisable

inversion du contrôle


- container prenant en charge l'exécution du code métier (composant)
- container assure lien avec la partie technique
- configurer plutôt que programmer
- approche framework vs librairie

injection de dépendances

- vers d'autres composants, vers des services techniques
- retirer du code métier la gestion des liens vers les autres composants

métiers

- faire gérer l'architecture applicative par le container

 Université Lille 1 Sciences et Technologies	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

CORBA-like (RMI ou CORBA)

Avantages des environnements à composants par rapport à CORBA-like


- configuration
- déploiement
- empaquetage (packaging)
- assemblage
- dynamicité
- gestion des interactions et des dépendances

Inconvénients de CORBA-like par rapport aux environnements à composants

- mélange code fonctionnel – code non fonctionnel
code métier "noyé" dans un ensemble d'appels à des services techniques
sécurité, transaction, persistance des données, annuaire, ...
- difficile à concevoir, comprendre, réutiliser

Différences entre RMI et CORBA

- RMI est propriétaire SUN / CORBA est une norme internationale
- RMI moins ambitieux que CORBA (RMI pour des petites architecture réseaux, sinon CORBA)
- RMI n'offre pas interopérabilité avec des objets d'autres langages (100% JAVA)
- RMI est beaucoup plus lent que les implantations de CORBA
- +RMI est une implantation standard et gratuite (incluse dans JDK)
- +RMI est object-oriented / CORBA est object-based
- +RMI est plus simple à mettre en oeuvre que CORBA
- +Pas besoins d'IDL : seulement l'interface de l'OD (Objet Distribué) et son URL

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0


+Les OD se manipulent comme les OL (grâce au DGC (Distributed Garbage Collector))

+RMI supporte le passage d'objets locaux par copie

+RMI permet une gestion de la sécurité (RMISecurityManager, ACL, ...)

Avantages de CORBA

- multi-OS, multi-langage
- métaphore du bus logiciel
- langage IDL

 Université Lille1 <small>Sciences et Technologies</small>	Conception d'applications réparties avancées	Examen CARA
Latest update : 25/03/2013	Copie étudiant	Current doc version : 1.0

EJB

4 services fournis par le serveur au container EJB :

- persistance (JPA)
- transaction (JTA)
- nommage (JNDI)
- sécurité (JASS)
- communication bas niveau (RMI)

Avantages

- on se focalise sur la logique applicative
- les services systèmes sont fournis par le container
- la logique de présentation est du ressort du client

Inconvénients

- trop compliqué, lourd, contraignant
- concepts objets (héritage, typage, polymorphisme, ...) difficilement exploitables
- mapping objet/relationnel limité