Architecture

Table of contents

1. Architecture générale

Voici une vue schématique de l'architecture globale et de l'interaction entre les éléments :

Architecture globale

La persistence se décline en :

- une partie LDAP : annuaire dans lequel sont stocké les agents, les queues Workflow, les modules de l'application cliente et les règles d'appartenances entre ces éléments.
- une Base de Données où sont stockés les contenus des emails, les tickets ...
- Le système de fichier dans lequel sont enregistrées les pièces jointes : les tailles de pièces jointes peuvent êtres importantes, on choisit de ne pas surcharger la BDD avec leur contenu. Seules sont stockées en base des références vers celles-ci et leur chemin dans le sysème de fichiers.

L'application J2EE est constituée de :

- une partir MOM (Middleware Orienté Message) qui gère les flux d'email en entrée (POP3/IMAP vers Application Cliente) et en sortie (Application Cliente vers SMTP).
- un ensemble d'EJB (Stateless Session Bean) dont le rôle est de gérer les intéractions entre le MOM, la persistance et la partie cliente.

L'application cliente se voit dotée de :

- le Core, le noyau de l'application : il identifie l'utilisateur et 'monte' les modules auquel cet utilisateur a accès (ces permissions proviennent du LDAP).
- un ensemble de modules : chaque module dialogue avec un ou plusieurs EJB Session (sans état) déployé sur le server J2EE.

2. Le MOM

Ci dessous, une représentaion des flux en jeux dans le MOM :

Circulation des flux dans le MOM

2.1. Les Queues JMS

Voici les différentes queues/topic JMS en jeu dans le système :

- incomingEmailQueue : c'est la queue des mail qui ont été acceptés par le Retriever. Ces emails sont destinés au Digester
- openedTicketQueue : transporte des tickets ouverts à destination du InputProcessor
- workflowQueue : transporte des item de travail (une réponse à apporter à une requête, une validation à effectuer ...). Les listeners de cette queue sont :

    - ResponseBuilder SSB associé à l'interface cliente
    - VirtualAgent : un agent virtuel
- validationQueue : après traitement par un agent, un item de travail est dirigé vers cette queue pour être analyser par le OutputProcessor
- outgoingEmailQueue : on y envoie des email devant être envoyés par SMTP. C'est EmailSender qui s'en chargera.
- eventTopic : chaque type d'évènement devant être stoqué (ou exporté vers autre système) est envoyé vers ce topic.

Note:

Par défaut, EmisEventReceiver est le seul MDB écoutant sur le eventTopic. Il est chargé de stocker chaque evênement dans une table dédiée de la BDD. Vous pouvez développer votre propre implémentation de EventReceiver afin d'informer votre système d'information lorsqu'un un type d'evenênement particulier est rencontré (arrivée d'un nouvel email, envoi d'un mail, loggin d'un agent...). Bien entendu, dans votre implémentation, vous pouvez utiliser la méthode la plus adaptée à votre environnement : JDBC, HTTP, ou même envoi d'un email.

2.2. Le Retriever

Le Retriever est un SSB managé par un MBean, lui même schédulé. Son rôle est de se connecter sur le(s) serveur(s) POP3/IMAP. Lorsqu'il obtient un email, il le passe dans une serie de Filter. Chaque Filter peut :

- analyser le contenu de l'email (destinataire, expéditeur, sujet, content, header ...) et décider si le mail est accepté pour intégration ou non.
- définir une réponse automatique à renvoyer à l'expéditeur.

Dès qu'un email est refusé, les autres Filter ne sont pas appliqués.

Si le mail est accepté, il est inséré dans la table email de la base de données, puis un object proxy est envoyé dans la queue incomingEmailQueue

Note:

Ce fonctionnement peut permettre de refuser un type de mail particulier selon un critère défini en envoyant une réponse de refus. Cela peut être utile pour définir une liste noire d'expéditeurs par exemple. L'interêt est que ces emails refusés ne viendront pas surcharger la base de données. Toutefois, dans la mesure du possible, il sera préférable de définir une politique de filtrage (anti-spam, black-list ...) en amont du système (MTA).

Le Retriever peut envoyer des messages sur :

- la queue incomingEmailQueue pour intégration.
- la queue outGoingEmailQueue pour l'envoi des réponses automatiques.
- le topic eventTopic pour l'envoi d'évènements.

2.3. Le Digester

Il est chargé d'intégrer reèlement les emails dans le système. Lorsqu'il recoit un nouveau message :

- il parse le sujet du mail pour y detecter l'éventuelle présence d'un identifiant de ticket sous la forme [#1234]. Si le ticket existe, il est réouvert et le mail y est associé. Si un identifiant de ticket n'est pas détecté, un nouveau ticket ouvert est créé.
- il interrroge la base de donnée pour récupérer l'identifiant du customer (sur la base de l'adresse email de l'expéditeur).
- si le customer n'est pas trouvé, le Digester interroge votre système d'information pour obtenir l'identifiant du client. Si celui n'existe pas dans la table customer, une nouvelle entrée est alors créée.

A la sortie du Digester, on ne parle plus d'email mais de ticket.

2.4. Le InputProcessor

Son but est de fournir un système modulaire et 'hotpluggable' d'application de règles. Les règles ou InputRule sont appliquées séquenciellement. Chaque InputRule après avoir analysé le ticket doit définir :

- si les règles suivantes doivent être appliquées.
- une éventuelle réponse automatique à envoyer.
- le nom de la file de workflow vers laquelle ce ticket va être envoyé.
- l'état du ticket (ouvert, fermé).
- la priorité du ticket.

A la sortie du traitement, un message est envoyé dans la workflowQueue pour traitement.