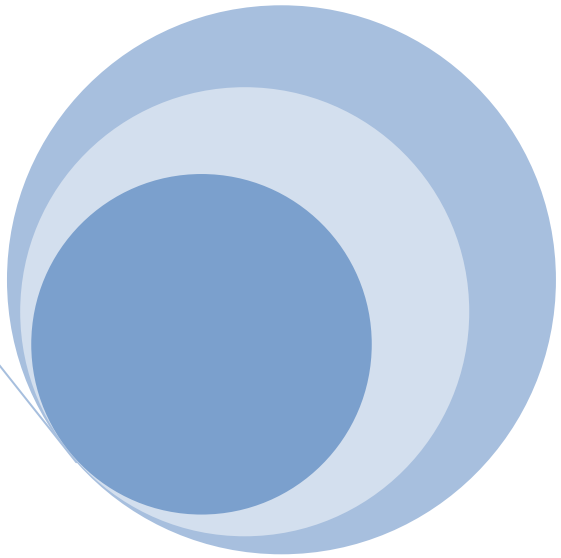
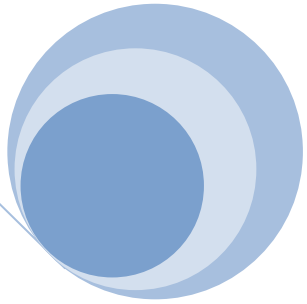
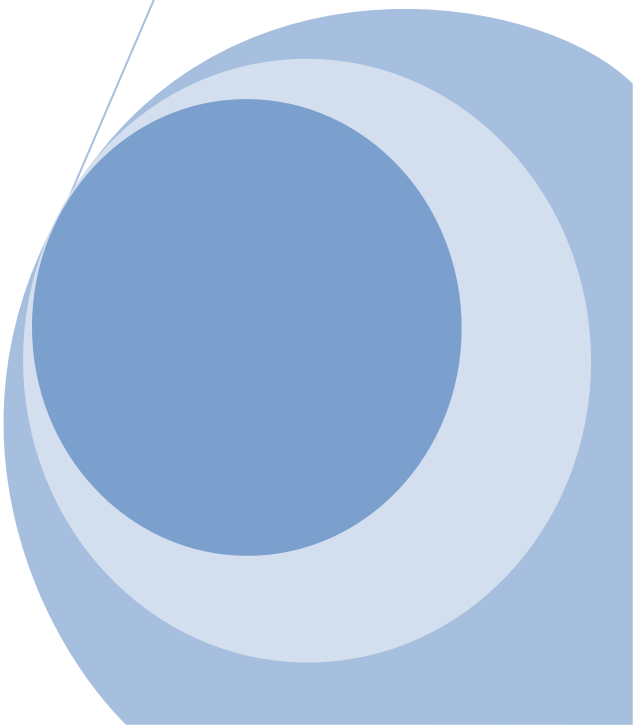

Relatório Projecto #4

Application Integration with Message Queues

Francisco Ferreira, João Lopes e Marco Simões

20-Dezembro-2009

Introdução

Neste projecto foi-nos pedido que elaborássemos uma aplicação distribuída que usasse um sistema de mensagens para integração dos seus diferentes módulos.

O objectivo é integrar as aplicações desenvolvidas no primeiro projecto prático da cadeira, isto é, as aplicações CameraSearchXML, CameraSumaryXML e CameraListBeautifier, e desenvolver uma nova interface que pede ao utilizador uma marca de câmara e utiliza as aplicações acima referidas para gerar uma resposta em XML com o sumário das câmaras e uma em HTML com a respectiva lista.

Estrutura da Aplicação

Com o intuito de garantir a interoperabilidade da nossa aplicação, utilizámos o serviço JMS (Java Messaging Service) sobre Glassfish. A utilização de MSMQ (Microsoft Messaging Queue) limitar-nos-ia aos sistemas operativos Microsoft, pelo que optámos por um serviço Java que nos mantém dependentes apenas da plataforma Java, presente em quase todos os sistemas de hoje.

Assim, adaptámos todas as nossas aplicações da primeira meta para funcionarem num sistema de mensagens: Cada uma cria as ligações necessárias par a sua inbox e outbox, e prepara-se para executar cada vez que uma mensagem chega à sua inbox. Quando tal acontece, esta é tratada, as execuções necessárias são efectuadas e o output é gerado e colocado na outbox.

Feitas estas adaptações, foi necessário criar uma aplicação para comunicação com o utilizador. Esta comunicação é simples, uma vez a ênfase do projecto se baseia no core do sistema e não na interface com o utilizador. Assim, esta aplicação apenas tem como funções pedir ao utilizador uma marca a pesquisar, dar-lhe conhecimento de qualquer tipo de erro ou sucesso e, finalmente, guardar os resultados em ficheiro.

Por fim, para gerir as comunicações entre as queues, desenvolvemos uma última aplicação que trata da gestão da lógica do sistema. Assim, desenvolvemos o myOrchestrator, que basicamente desempenha as seguintes funções:

- Reage a uma mensagem de pesquisa colocada na outbox da aplicação do utilizador e coloca-a na inbox da aplicação CameraSearchXML;

- Reage a uma mensagem colocada na outbox do CameraSearchXML e faz uma das seguintes:

- Se a mensagem representar um erro, reencaminha-a para a inbox do utilizador;

- Se a mensagem for um resultado correcto, é encaminhado para a inbox do CameraSummaryXML e do CameraListBeautifier;

- Reage a uma mensagem colocada na outbox do CameraSummaryXML e do CameraListBeautifier. Quando uma mensagem com o mesmo id aparece nestas duas outboxes, é criada uma objectMessage que junta o conteúdo destas duas e é colocada na inbox do utilizador. Nenhum sistema de ordem é considerado nesta parte, sendo tudo assíncrono. Para tal usamos uma hashtable para guardar a primeira mensagem que esteja disponível, sendo que a segunda cria o objecto e final e envia-o para o utilizador.

Camera List Beautifier

Para transformar o XML em HTML foi utilizado uma biblioteca existente no java SE. O HTML gerado faz uso de javascript e CSS para tornar o conteúdo mais dinâmico e agradável ao utilizador. Assim carrega-se o XML a partir de duas StreamSources. Uma para o XSL e outra para o XML. No entanto o javascript e CSS visto serem dados estáticos sempre iguais para qualquer html gerado, têm de ser colocados a par com o output do HTML.

É utilizado uma instanciação de TransformerFactory e outra de Transformer existentes na package javax.xml.transform para gerar o HTML. O TransformerFactory recebe o XSL, ficando à "espera" de um XML para ser transformado em HTML.

Finalmente o HTML gerado é encaminhado para o utilizador, através do sistema de comunicação do sistema, neste caso, filas de mensagens

Tempo Dispendido

A tarefa que requereu mais tempo foi a de pesquisa e colocação do sistema a comunicar com as filas do Glassfish. A implementação dos mecanismos entre os vários processos foi relativamente simples e rápida.

A semelhança dos trabalhos anteriores, algum tempo de pesquisa e leitura de manuais/tutoriais não foi incluído, pela dificuldade de medir este tempo.

Francisco Ferreira

8 horas na implementação do CameraListBeautifier.

2 horas na relatório

João Lopes

12 horas na implementação na construção de filas, myOrchestrator e UserInterface.

1 hora no relatório

Marco Simões

8 horas na adaptação do CameraSearchXML e CameraSummaryXML.

3 horas no relatório

Conclusão

O maior problema encontrado foi colocar o sistema a encontrar e a utilizar as filas do Glassfish.

Com este trabalho foi possível observar a utilidade de filas de mensagens para a comunicação entre processos diferentes, que poderão estar a decorrer em máquinas geologicamente separadas. Assim como o uso benéfico de filas com transacções.

20 de Dezembro de 2009

Francisco Ferreira
fmsf@student.dei.uc.pt
2006124182

João Lopes
jmlopes@student.dei.uc.pt
2006125131

Marco Simões
msimoes@student.dei.uc.pt
2006125287

Integração de Sistemas
Mestrado em Engenharia Informática no
Departamento de Engenharia Informática da
Faculdade de Ciências e Tecnologia da Universidade de Coimbra