Construção de Sistemas de Software

SaleSys - Web e SOAP

Grupo 018
João Rodrigues, nº 45582
António Rodrigues, nº 40853
Simão Neves, nº 45681

Índice

Introdução

Nesta fase do projecto foram desenvolvidas duas versões do sistema SaleSys. Estas versões foram Web e SOAP. Recorremos ao WildFly enquanto servidor, e como tal pudémos delegar algumas responsabilidades ao container, nomeadamente a instanciação de objetos, a gestão de entidades JPA, e assim tirar partido do JTA, que nos permite fazer uma gestão de transações de forma facilitada e eficiente. As duas versões desenvolvidas assentam sobre a mesma camada de negócio e consequentemente de persistência, conforme será em seguida detalhadamente descrito.

Business

Nesta camada é desenvolvida a lógica aplicacional e são tomadas as decisões “mais importantes”. No caso da aplicação SaleSys, tivemos que definir entidades, assim como as suas associações, de modo a que estas integrassem a já existente aplicação. Para cada caso de uso foi criado um handler. Handler este que é exposto como web service para que possa ser acedido remotamente por clientes SOAP. Quanto aos beans de sessão, optámos por ter apenas beans sem estado (Stateless). Isto porque na aplicação SaleSys não existe a necessidade de manter a sessão entre vários requests ao servidor de um mesmo cliente.

Web

No desenvolvimento de um cliente web tirámos o máximo partido do que já existia na camada de negócio. Esta camada de apresentação recorre a jsp’s para o display das views, que por sua vez recorrem ao respectivo model para aceder aos dados que necessitam. O referido model é disponibilizado para a view na forma de um bean que é definido como atributo do request, deste modo conseguimos que este seja apenas utilizado num e num só request dado o seu life cycle. Para cada caso de uso foi criada a respectiva View (jsp’s), o respectivo controller (Action) e o seu Model. Seguindo assim o padrão MVC, Model-View-Controller.

SOAP

Na camada de apresentação do cliente de texto é utilizado o protocolo SOAP, Simple Object Access Protocol. Este protocolo é usado para troca de informações, objetos ou não, e assenta sob o protocolo HTTP (mas podia ser outro). Foram desenvolvidos alguns Handlers na camada de negócio, que são expostos como WebServices, que podem ser acedidos remotamente através de clientes SOAP. Para cada Service End Point (Handler neste caso) foi necessário fazer o import do wsdl (Web Service Description Language) correspondente (criado pelo servidor WildFly) que criou stubs prontos a serem utilizados na aplicação cliente SOAP.
O cliente utiliza os stubs para comunicar com o servidor utilizando o protocolo correcto. Além disso, foi preciso anotar a interface que os Handlers implementam com @Remote, que permite que os métodos incluídos neste interface sejam acedidos por uma JVM não local.
Notámos que ao fazermos import dos wsdl e com base na nossa estrutura de pastas (packages) houve repetição de algumas classes geradas, existe por exemplo múltiplas AplicationException_Exception em packages diferentes no cliente SOAP, porque exportamos individualmente e cada um deles vai gerar as suas depêndencias.

Conclusão

Tendo este projecto como primeira abordagem a servidores em Java e à implementação dos respectivos clientes, neste caso foram Web e SOAP, tivemos a oportunidade de fazer uso da inversão de controlo, delegando algumas responsabilidades ao Container. Pudémos comprovar algumas das vantagens da utilização do Maven, como por exemplo a gestão de dependências e a organização de projetos por módulos. Verificámos que o JTA remove imensa carga de trabalho no que diz respeito a transações, visto que este garante a atomicidade como seu comportamento padrão o que em muitos casos é imprescindivel.
No caso do cliente Web, o uso de MVC é bastante explicito sendo que para cada view existe um controller que em seguida disponibiliza à view um model para que esta tenha acesso a determinados dados considerados relevantes aquando da sua apresentação.
No desenvolvimento de um cliente SOAP, deparámo-nos com algumas limitações, nomeadamente a existência de ciclos aquando da chamada a uma função remota, ou seja, se uma Sale tem como um dos seus atributos uma lista de Transactions e cada uma destas tem como atributo uma Sale, então existe circularidade, o que levanta uma Exception. Esta Exception foi resolvida inserindo a anotação @JoinColumn na lista de Transactions da Sale, e removendo o atributo Sale da entidade Transaction.