04

Conhecendo o Sticky Session

Transcrição

O nosso problema do login tem a sua causa no balanceamento das requisições. Quando um servidor Tomcat recebe um cookie com uma identificação que ele não conhece, automaticamente vai criar um novo cookie e apagar a identificação antiga. Nunca podemos nos logar :(

Session Migration

Para resolver isso, existem duas estratégias: Quando um servidor recebe a requisição e não possui a sessão identificada pelo Cookie que foi passado, ele sabe que um dos outros servidores Tomcat pode ter. E caso isso seja verdade, esse outro servidor irá **migrar a sessão e todo seu estado** (`HttpSession` e os atributos) para o primeiro servidor. Após isso a sessão e seus estados passam a **existir apenas na primeira máquina e não mais na segunda**. Essa abordagem é conhecida como *Session Migration* e se encontra na especificação da *Servlet API*.

Sticky Session

A segunda estratégia é focada no balanceador de carga da Amazon e essa vamos seguir, pois não queremos usar algo específico do servidor ou plataforma de desenvolvimento. Como vimos, o problema está no balanceamento das requisições, então vamos parar com isso e garantir que um cliente (navegador) sempre converse com o mesmo servidor Tomcat! Não vamos mais balancear por requisição e sim por cliente! Na primeira requisição do cliente, o balanceador vai escolher uma instância, mas a partir daí todas as requisições seguintes desse cliente serão enviadas para a instância escolhida. Essa estratégia se chama de ***Sticky Session*** ou ***Session Affinity*** e se baseia em um novo cookie, que o balanceador de cara vai criar. Nesse cookie fica a informação de qual das duas máquinas o balanceador escolheu. Ou seja, em cada requisição o balanceador vai verificar esse novo cookie, para saber para qual instância a requisição deve ser roteada.