When to use Stateful session bean over Stateless session bean?

Stateful session bean is defined as follow:

> Stateful Session Beans The state of an object consists of the values of its instance variables. In a stateful session bean, the instance variables represent the state of a unique client-bean session. Because the client interacts ("talks") with its bean, this state is often called the conversational state.

Stateless session bean is defined as follow:

> Stateless Session Beans A stateless session bean does not maintain a conversational state with the client. When a client invokes the methods of a stateless bean, the bean's instance variables may contain a state specific to that client, but only for the duration of the invocation. When the method is finished, the client-specific state should not be retained. Clients may, however, change the state of instance variables in pooled stateless beans, and this state is held over to the next invocation of the pooled stateless bean. Except during method invocation, all instances of a stateless bean are equivalent, allowing the EJB container to assign an instance to any client. That is, the state of a stateless session bean should apply accross all clients.

It is also mentioned the advantage of stateless session bean as follow:

> Because stateless session beans can support multiple clients, they can offer better scalability for applications that require large numbers of clients. Typically, an application requires fewer stateless session beans than stateful session beans to support the same number of clients.

So the question that comes to mind is when one should use stateful session beans? To my naive understanding of the matter, one should stick to use stateless session bean as he can.

What would be the candidates in which one should use stateful session bean? Any good examples?

Session Bean

ejb stateless-session-bean stateful-session-bean session-bean

asked Aug 1 '13 at 17:13
sheidaei
2,129 10 30 56

Related: stackoverflow.com/questions/8887140/… – BalusC Dec 27 '14 at 10:09

2 Answers

First you have to understand how the beans are created and handled on the server.

For **stateless session beans** the server can maintain a variable amount of instances in a pool. Each time a client requests such a stateless bean (e.g. through a method) a random instance is chosen to serve that request. That means if the client does two subsequent requests it is possible that two different instances of the stateless bean serve the requests. In fact there is no conversational state between the two requests. Also if the client disappears, the stateless bean does not get destroyed and can serve the next request from another client.

On the other hand a **stateful session bean** is closely connected to the client. Each instance is created and bounded to a single client and serves only requests from that particular client. So happens that if you do two subsequent requests on a stateful bean, your request will be served always from the same instance of the bean. That means you can maintain a conversational state between the requests. At the end of the lifecyle the client calls a remove method and the bean is being destroyed/ready for garbage collection.

So when to use stateless or stateful?

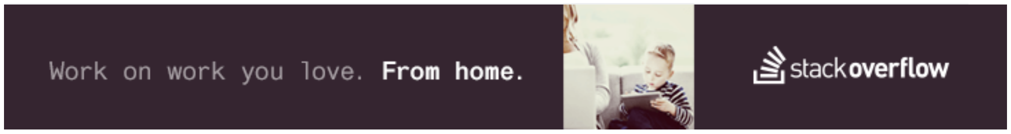
That mainly depends on whether you want to maintain the **conversational state**. For example if you have a method that adds up to numbers and return the result you use a stateless bean because its a one time operation. If you call this method a second time with other numbers you are not interested in the result of the previous addition anymore.

But if you want for example count the number of requests a client has done, you have to use a stateful bean. In this scenario it is important to know how often the client has requested the bean method before, so you have to maintain conversational state in the bean (e.g. with a variable). If you would use a stateless bean here the request of the client would be served each time from a different bean what messes up your results.

edited Jan 9 '15 at 7:34 answered Sep 20 '13 at 12:47
tobiasdenzler
515 7 10

7 "*If the clients disappears the bean get destroyed too*". Actually, stateful session beans do not get destroyed automatically unless a method decorated by `@Remove` (`javax.ejb`) is invoked explicitly (that method need not even be coded. It could simply be left empty/blank given that it is annotated by `@Remove`). If the associated client forgot to destroy a stateful session bean, that bean would be kept on dangling on the server until the container itself decides to remove it using its own policy. Am I going wrong? – Tiny Dec 26 '14 at 5:09

1 Of course you are right. More information on bean lifecycle can found here: docs.oracle.com/javaee/6/tutorial/doc/giplj.html – tobiasdenzler Jan 9 '15 at 7:31



I think that the greatest example of using a **Stateful session bean** is for a *Shopping Cart*, where you store all products which user wants to buy.

edited Dec 3 '14 at 10:35 answered Jun 24 '14 at 5:13
snajahi Beezy
368 2 6 20 **587** 5 17

protected by BalusC Dec 27 '14 at 10:07

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?