AG Heterogene Informationssysteme
Prof. Dr.-Ing. Stefan Deßloch
Fachbereich Informatik
Technische Universität Kaiserslautern

Middleware for Heterogeneous and Distributed Information Systems – Exercise Sheet 3

Wednesday, November 12, 2008 – 10:00 to 11:30 – Room 48-379

Application server middleware

In the previous exercise, database gateways have been addressed. In this exercise, we discuss database access and transaction processing using application server middleware.

Let's first look at database access by means of a JDBC database gateway again. The system architecture is depicted in Figure 1. The application program sends requests to the JDBC driver. These requests are then translated and sent to the database, possibly across the network. The database response is received by the JDBC driver, translated, and passed on to the application program.

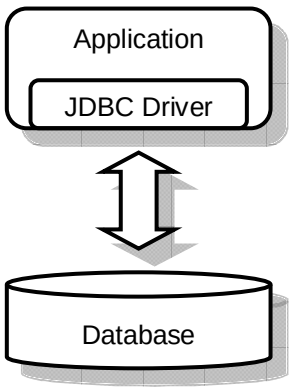

Figure 1: Direct database access using a JDBC driver

1. The situation changes when *application server middleware* enters the picture. Depict the system architecture when application server middleware is used! What are the differences compared to the system architecture in Figure 1?
2. Say, we use a J2EE application server as application server middleware. What configuration steps are required to enable access to the data sources by the application program[1]?

[1] See the lecture notes, chapter 3 on database gateways, slide "JDBC DataSource"

3. How does the application program create a connection to the database using a JDBC database gateway? How are database connections obtained from an application server?
4. How does the application program demarcate transactions using a JDBC database gateway? How does (explicit) transaction demarcation work using an application server? Explain why different interfaces are used in either case!
5. What advantages do application servers offer with respect to database access and transaction processing? Do you see any drawbacks?

Transaction Processing Monitors

Transaction processing monitors (TP monitors) are regarded as one of the oldest form of middleware. Roughly speaking, TP monitors provide a programming environment for distributed transactional applications. In chapter 4 "Remote Procedure Calls and Distributed Transactions" the term transaction manager (TA manager, TM is used in the X/Open DTP standards) has been introduced. Briefly summarize the main functionality of TP monitors and TA managers. What are the differences?

Implicit Transaction Demarcation

The recommended way to manage transactions in J2EE is through container-managed transactions, i.e. the EJB container sets the boundaries of the transactions. As opposed to bean-managed transactions, no calls to the `UserTransaction` interface are required. In fact, the transactional characteristics of enterprise bean methods are specified in the deployment descriptor by means of so called *transaction attributes*. Thus, the transactional characteristics can easily be changed without any modifications of the developed code.

A transaction attribute controls the scope of a transaction, i.e. it controls whether an enterprise bean's method executes within the caller's transaction. A transaction attribute can take six different values: `Required`, `RequiresNew`, `Mandatory`, `NotSupported`, `Supports`, and `Never`.

Describe the meaning of each of the transaction attributes[2]! Give meaningful examples!

[2] See *J2EE Transaction Frameworks, Part 3* available at http://www.onjava.com/pub/a/onjava/2001/06/06/j2ee_trans.html

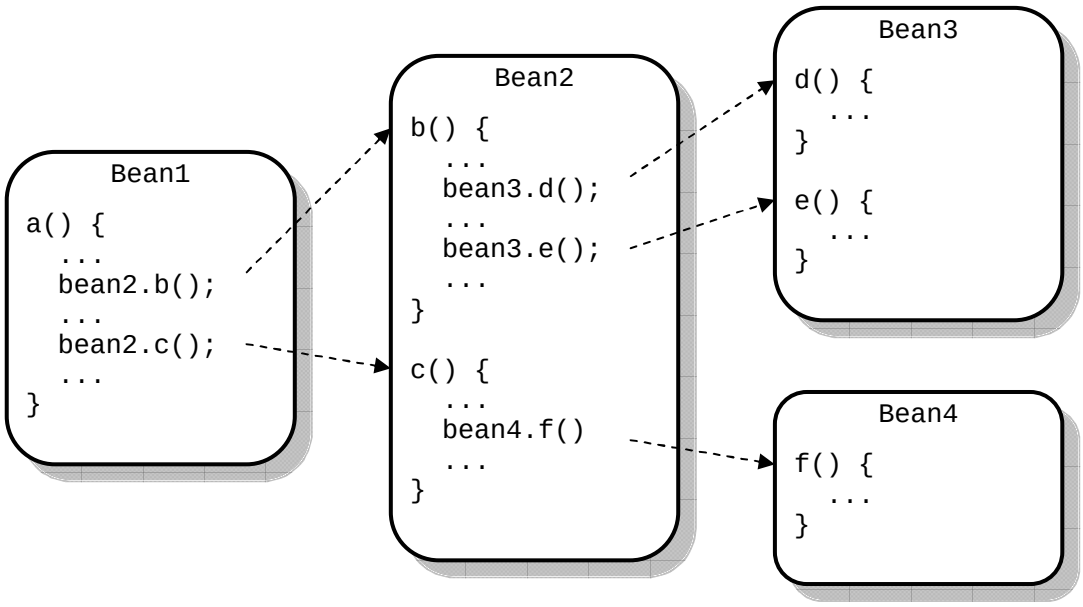

Figure 2: Sample enterprise bean methods

Consider the enterprise bean methods depicted in Figure 2. Assume that `bean1.a()` is initially called outside an transaction scope. What transaction scopes emerge for the following transaction attribute settings?

Required	RequiresNew	Mandatory	NotSupported	Supports	Never
bean2.b() bean2.c() bean3.e()	bean1.a()	bean3.d()		bean4.f()	
bean1.a() bean2.b()		bean3.e()	bean3.d()	bean2.c()	bean4.f()
bean1.a() bean3.d() bean4.f()	bean2.b() bean3.e()		bean2.c()		