Deep Dive into Oracle ADF Transactions: Advanced Techniques

Eugene Fedorenko
eProseed NL
The Netherlands

Keywords:

ADF, Transaction, Deep dive, DB Transaction, Data Control Frame, Data Control Scope, Taskflow transaction option

Introduction

What is a transaction all about? Let's consider a transaction as a unit of work that must be either entirely completed or aborted. Any transaction is consistent which means that it brings a system from one valid state to another. So, even I case of failure we don't get stuck half way. A classic example of a transaction comes from business world and is about transferring money between two bank accounts. Let's say we have taken money from the first account and got an error before putting money to the second account. So, the money seem to be nowhere. The concept of transaction makes this awkward situation impossible and the entire unit-of-work is going to be cancelled returning money back to the first account. Another valuable feature of a transaction is isolation. Which means that while a transaction is being processed other users don't see any results of its work until the whole transaction, the entire unit of work is completed. So, we don't confuse the other user sessions with unfinished changes. On the other hand, we and we only may see all the changes that we've done since the start of the transaction. This fact means that any system capable of transaction management should be aware of state management as well. We should be able to track all changes that we have done in the transaction in order to be able to either commit or rollback all of them.

Transactions in ADF

Oracle ADF as a matured framework for enterprise applications just must support such essential thing as a transaction. The question is what parts of the framework, what components are in charge of transaction management. Oracle ADF architecture is built on top of different layers. Transaction management in Oracle ADF is not implemented in one single place, it is spread across the framework. Basically, ADF works with transactions at the business service layer, model layer and controller layer, assuming that we consider standard ADF technologies such as Business components, Data Controls, Bindings and Task flows.

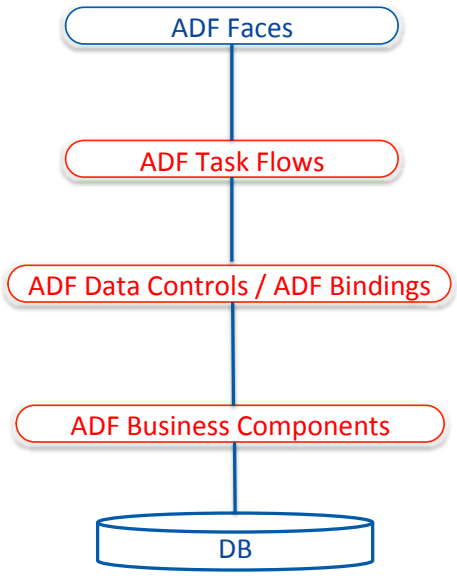

Illustration. 1: ADF layers with transaction management

ADF BC Transactions
The following diagram represents represents the core building blocks of ADF Business Components at run-time.

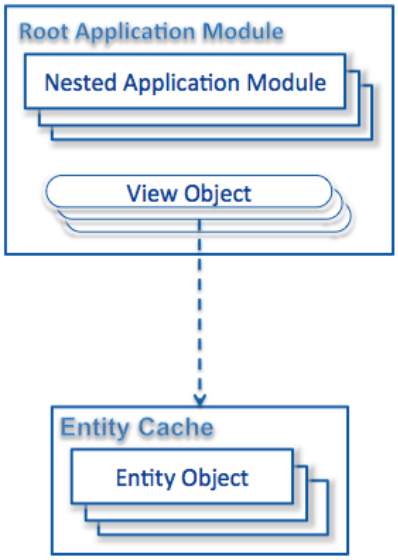

Illustration. 2: ADF BC core building blocks

There is an instance of a root application module containing view object instances. View object instances might be backed up by entity objects that are stored in entity collection or in other words entity cache. A root application module may also contain nested application modules which in their turn my contain their own view object instances. This is very important that all view object instances and nested application modules within a single root AM share the same entity cache. The question is How? ADF BC suppose that each user session has its own entity cache. So, what links my application module to my and only my entity cache? And here is where DB Transaction object comes to the scene:

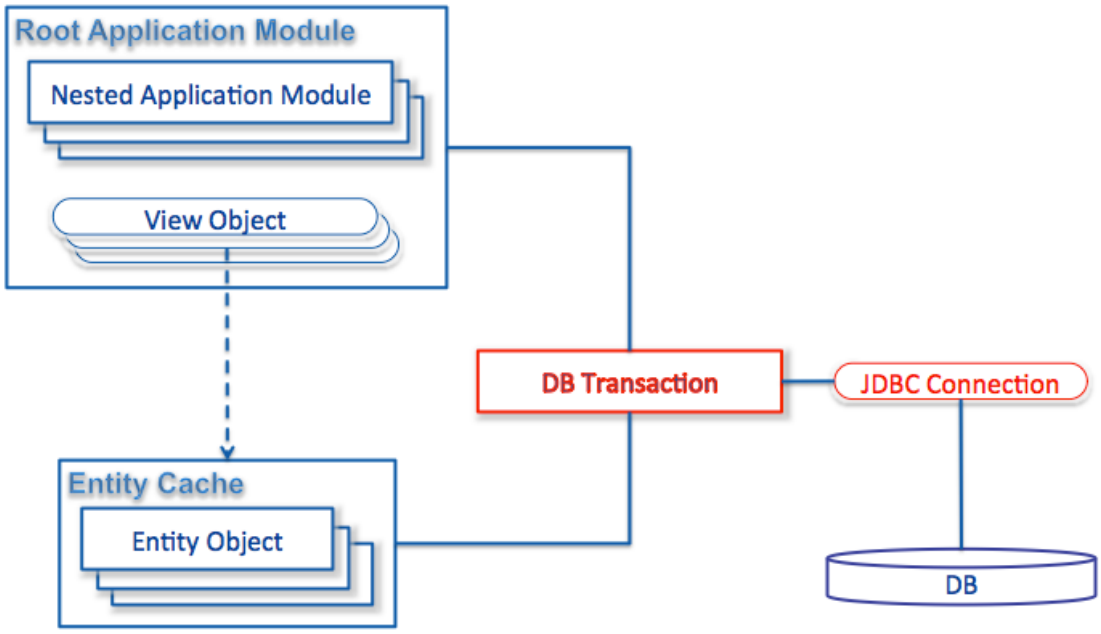

Illustration. 3: DB Transaction object

It is an internal framework object that actually contains entity cache and provides it to all application modules registered with this DB Transaction object. Furthermore DB Transaction object contains a DB connection and it provides all jdbc-related services such as creating and executing callable statements.

Shared Transactions

Many developers think that actually application module is responsible for containing entity cache, holding DB connection and interacting with database. That's not actually true. An application module is just attached to DB Transaction object consuming entity cache and DB connection from it. The word "attached" means that there could be many root application modules referring to the same DB Transaction object.

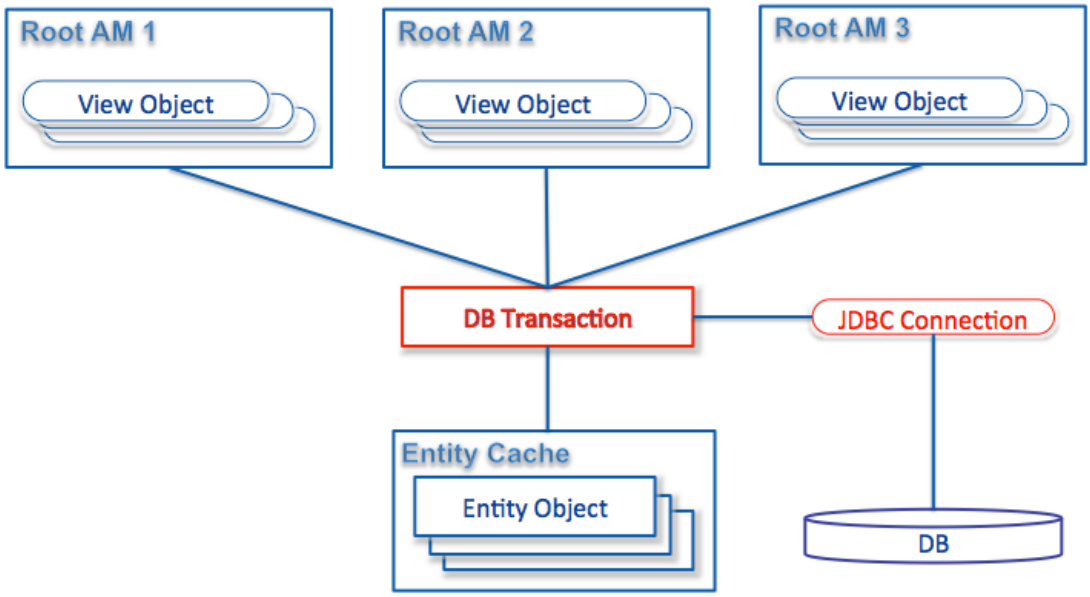

Illustration. 4: Shared Transaction

In that case the transaction is called "shared". Each application module attached to it consumes the same DB Connection and the same entity cache. There is a common myth that any instance of a root application module always requires a dedicated DB connection. Obviously, that's not always the case.

ADF Model Transactions

Oracle ADF introduces ADF model layer consisting of ADF Data Controls and ADF Bindings. Data controls act as proxy adaptors for the business services. In other words data controls invoke business services and return the result wrapped in a generic data structure to be used by ADF bindings. On the other hand ADF bindings is a declarative approach to bind UI components to the underlying data collections. And as long as data controls provide the data in generic data structures the UI components are abstracted away from the details of data access and details of invoking data control operations.

And when it comes to transaction management a data control is the one who is responsible for that. ADF provides out-of-the-box a number of various data controls for different business service technologies. However, not every data control type implements all aspects of transaction management as we know it. Even though all datacontrols are originally designed to carry state, so they meet that requirement very well, only few of them are actually able to explicitly commit and rollback transactions. Basically, only ADF BC, EJB and JPA-based bean datacontrols can explicitly commit and rollback changes out-of-the box.

Data Control Frames

In a real-life application data for UI components on a web page can be provided not only by a single but by a few data controls simultaneously and the real power of the transaction is that a set of operations can be committed or rolled back together. Oracle ADF uses data control frame for this job. This is an internal framework object which serves as a bucket for holding data controls used in the binding container. The data control frame refers to a transaction handler for managing all transaction related activities such as commit and rollback of transactions, working with savepoints. The transaction handler delegates all the calls to the respective data controls. In other words a data control frame is a bunch of data controls that belong to the same transaction.

ADF Controller Transactions

The controller layer in Oracle ADF is represented by Task flows. A task flow is a fundamental design artifact in ADF framework which actually lets developers consider ADF application as a set of business processes rather then a set of web pages. A task flow contains various activities and navigation rules between them. Task flows can be reused and coupled with other task flows to compose a large ADF application implementing the concept of modularization and reuse. So, from UI and controller perspective an application is just a bunch of task flows that are, actually, key players in transaction management in ADF application.

Each task flow implements some business logic representing some business process. It may require a separate transaction for that, or may be it is supposed to work within existing transaction. Some task flows share their data with the others, some of them prefer to keep things private. This behavior and actually how the task flow participates in transaction management is defined by task flow transaction options.

Data Control Scope

The “Share data controls with calling task flow” task flow property is essentially important and it is tightly coupled to the concept of data control frames. The value of this check box actually determines whether a new data control frame should be created or not.

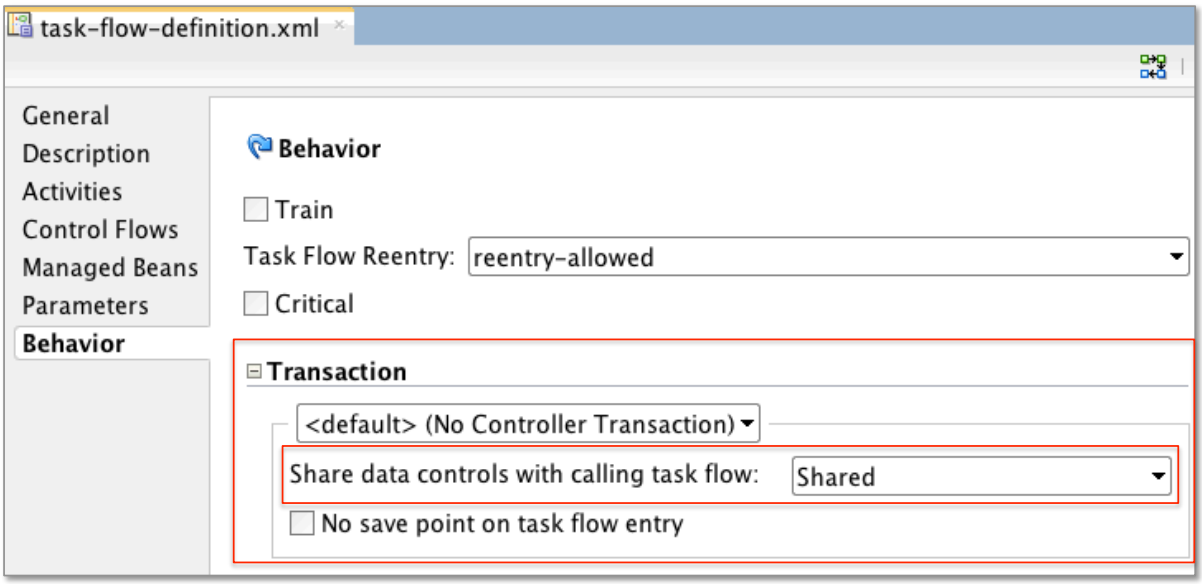

Illustration. 5: Data Control Scope

Shared Control Scope

If the data control scope is shared that means that the framework will attempt to reuse a datacontrol frame provided by the the calling task flow rather then creating a new one. Which means that the task flow will share with the caller the same instance of the datacotrol (if they are referring to the same name and type). In case of ADF BC data control it means that both task flows will use the same instance of application module providing the same VO instances. Furthermore, since the data control frame is shared the transaction handler is going to be shared as well which in case of ADF BC ends up with the same DB transaction object, providing the same DB connection and the same entity cache.

Isolated Control Scope

If the data control scope is isolated a brand new datacontrol frame will be created to serve this task flow. And this data control frame will keep all its stuff separately. Even though the task flow refers to the same data control definition as the caller does a separated instance of this datacontroll will be created. There will be separated instances of AM, VO, database connection and entity cache

Task Flow Transaction Options

Task flow transaction options allow us define where the transaction starts and finishes in terms of application pages and commit and rollback the associated data controls as a group.

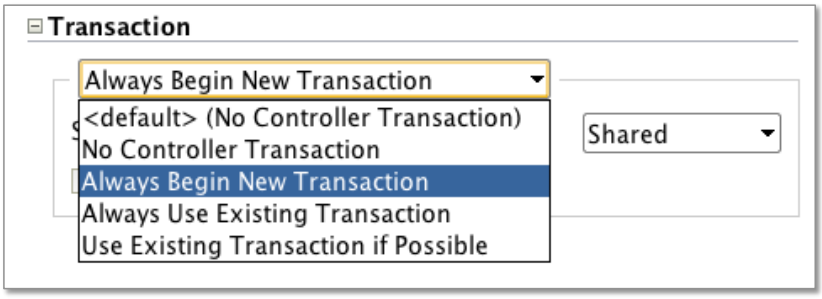

Illustration. 6: Task Flow Transaction Options

Basically, these options determine whether changes should be tracked within he task flow and what should be done with those changes when we exit the task flow. There are three available values for the controller transaction management: Always Begin New Transaction, Always Use Existing Transaction, Use Existing Transaction if Possible.

No Controller Transaction

The “No Controller Transaction” switches off controller transaction management by the task flow. Which means that the task flow does nothing with the transaction declaratively. It neither starts a new transaction nor creates a savepoint on task flow entry. And its return activities neither commit nor rollback the transaction. So, the task flow itself doesn’t care about transaction at all meaning that it is all in your hands and you should manage the transaction yourself if you need. The “No Controller Transaction” option could be useful when you need to move out of the boundaries of declarative settings to implement specific use cases. For example, consider a scenario where task flow's transaction setting depends on complex business conditions and the declarative approach fails to meet the requirement. In such cases, we can use the model APIs to manage a transaction programmatically.

Contact address:

Eugene Fedorenko
eProseed NL
Rembrandtlaan 24e
3723 BJ, Bilthoven

Phone: +31650420681
Blog: adfpractice-fedor.blogspot.com
Twitter: @fisbudo
Email eugene.fedorenko@eproseed.com
Internet: http://www.eproseed.com