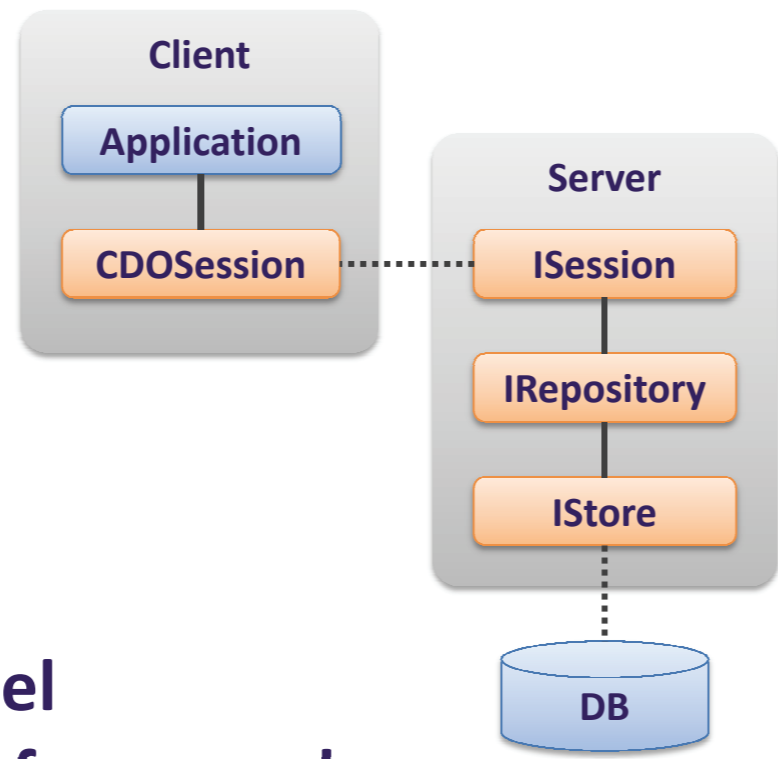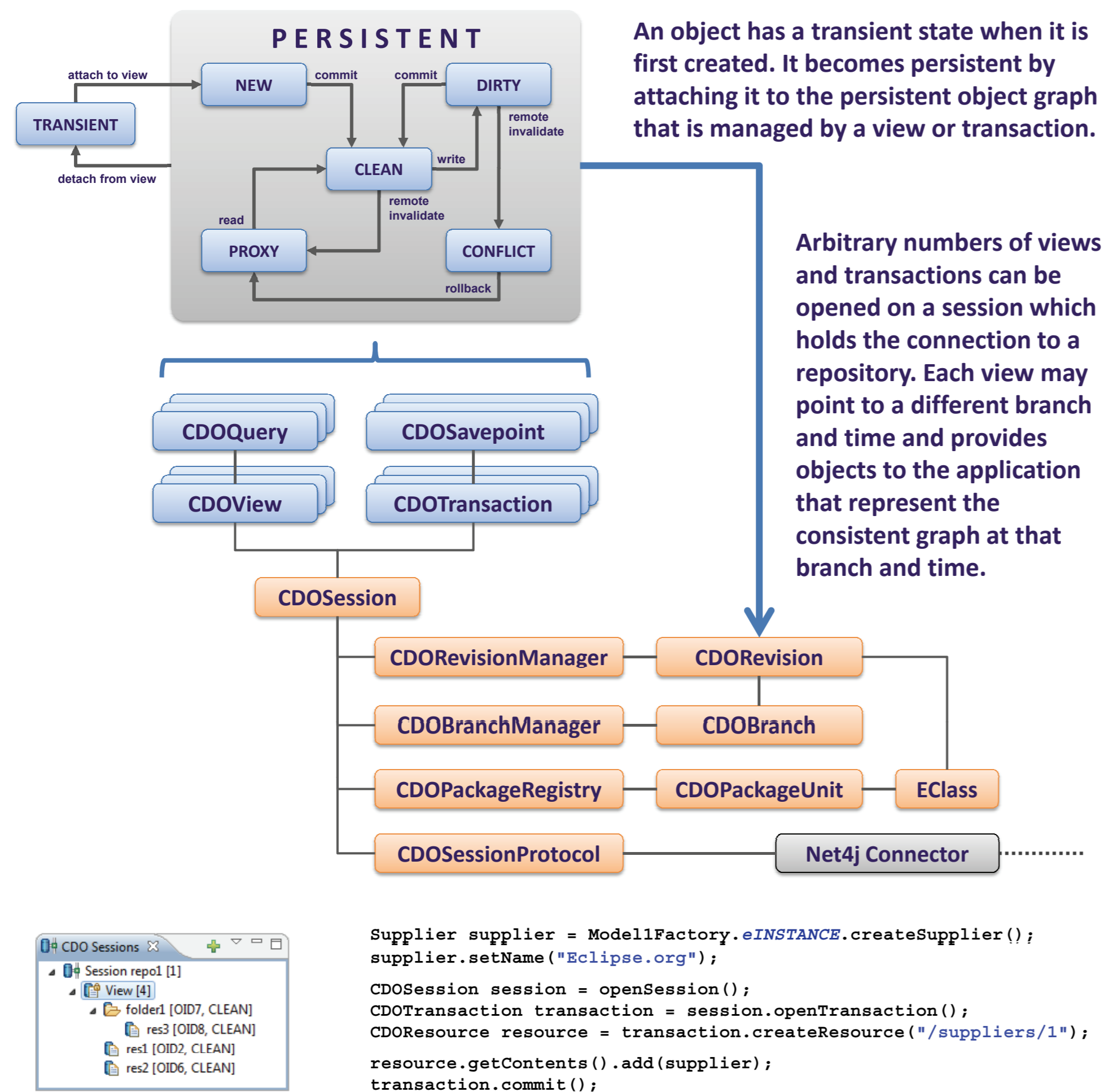# CDO
The Model Repository

CDO is both a development-time model repository and a run-time persistence framework. Being highly optimized it supports object graphs of arbitrary size. The storage back-end is pluggable and CDO offers transactions with save points, explicit locking, change notification, queries, temporality, branching, merging, offline and fail-over modes, …

Client
- Application
- CDOSession

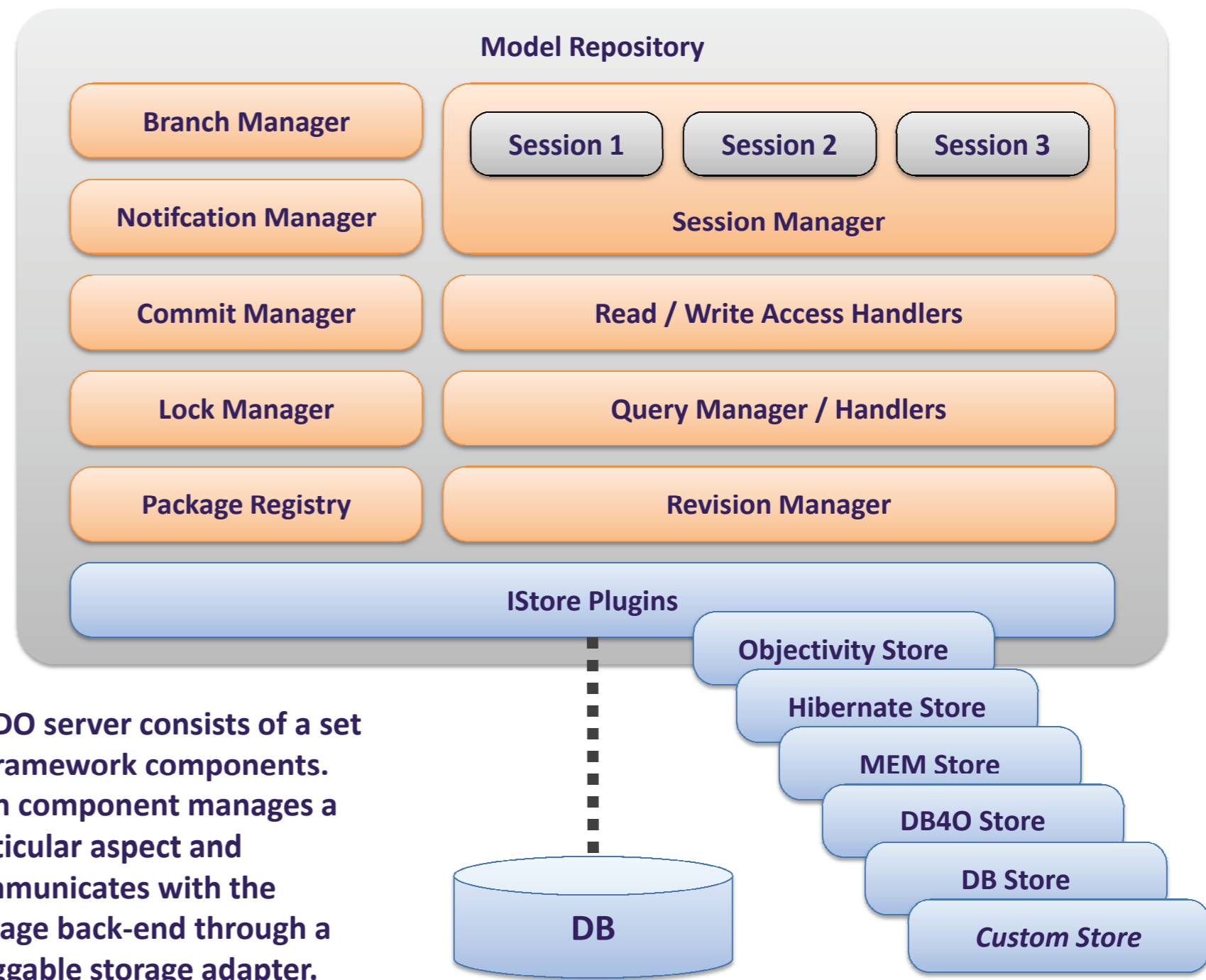**Server**
- ISession
- IRepository
- IStore
- DB

The Client

CDO is designed to work with the Eclipse Modeling Framework (EMF) and Ecore, a free Java™ implementation of the EMOF™ specification of the OMG.
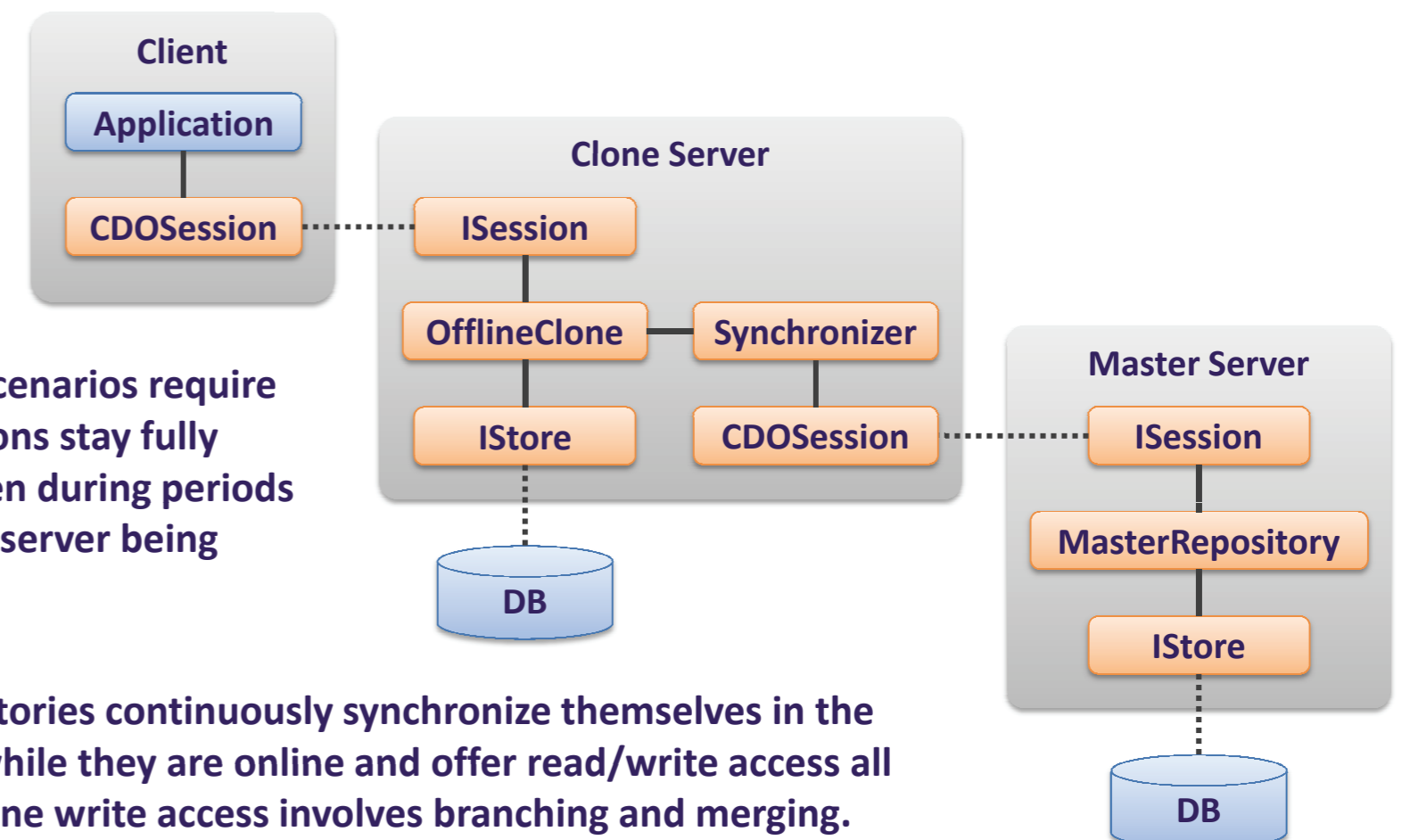
PERSISTENT

- TRANSIENT
- attach to view
- detach from view
- NEW — commit
- commit — DIRTY
- CLEAN — write
- remote invalidate
- read
- PROXY
- remote invalidate
- CONFLICT
- rollback

An object has a transient state when it is first created. It becomes persistent by attaching it to the persistent object graph that is managed by a view or transaction.

Arbitrary numbers of views and transactions can be opened on a session which holds the connection to a repository. Each view may point to a different branch and time and provides objects to the application that represent the consistent graph at that branch and time.

- CDOQuery
- CDOSavepoint
- CDOView
- CDOTransaction
- CDOSession
- CDORevisionManager — CDORevision
- CDOBranchManager — CDOBranch
- CDOPackageRegistry — CDOPackageUnit — EClass
- CDOSessionProtocol — Net4j Connector

```
CDO Sessions
▲ Session repo1 [1]
  ▲ View [4]
    ▲ folder1 [OID7, CLEAN]
       res3 [OID8, CLEAN]
    res1 [OID2, CLEAN]
    res2 [OID6, CLEAN]
```

```java
Supplier supplier = ModelFactory.eINSTANCE.createSupplier();
supplier.setName("Eclipse.org");

CDOSession session = openSession();
CDOTransaction transaction = session.openTransaction();
CDOResource resource = transaction.createResource("/suppliers/1");

resource.getContents().add(supplier);
transaction.commit();
```

The Server

Model Repository

- Branch Manager
- Notifcation Manager
- Commit Manager
- Lock Manager
- Package Registry
- Session 1 / Session 2 / Session 3
- Session Manager
- Read / Write Access Handlers
- Query Manager / Handlers
- Revision Manager
- IStore Plugins
  - Objectivity Store
  - Hibernate Store
  - MEM Store
  - DB4O Store
  - DB Store
  - Custom Store
- DB

A CDO server consists of a set of framework components. Each component manages a particular aspect and communicates with the storage back-end through a pluggable storage adapter.
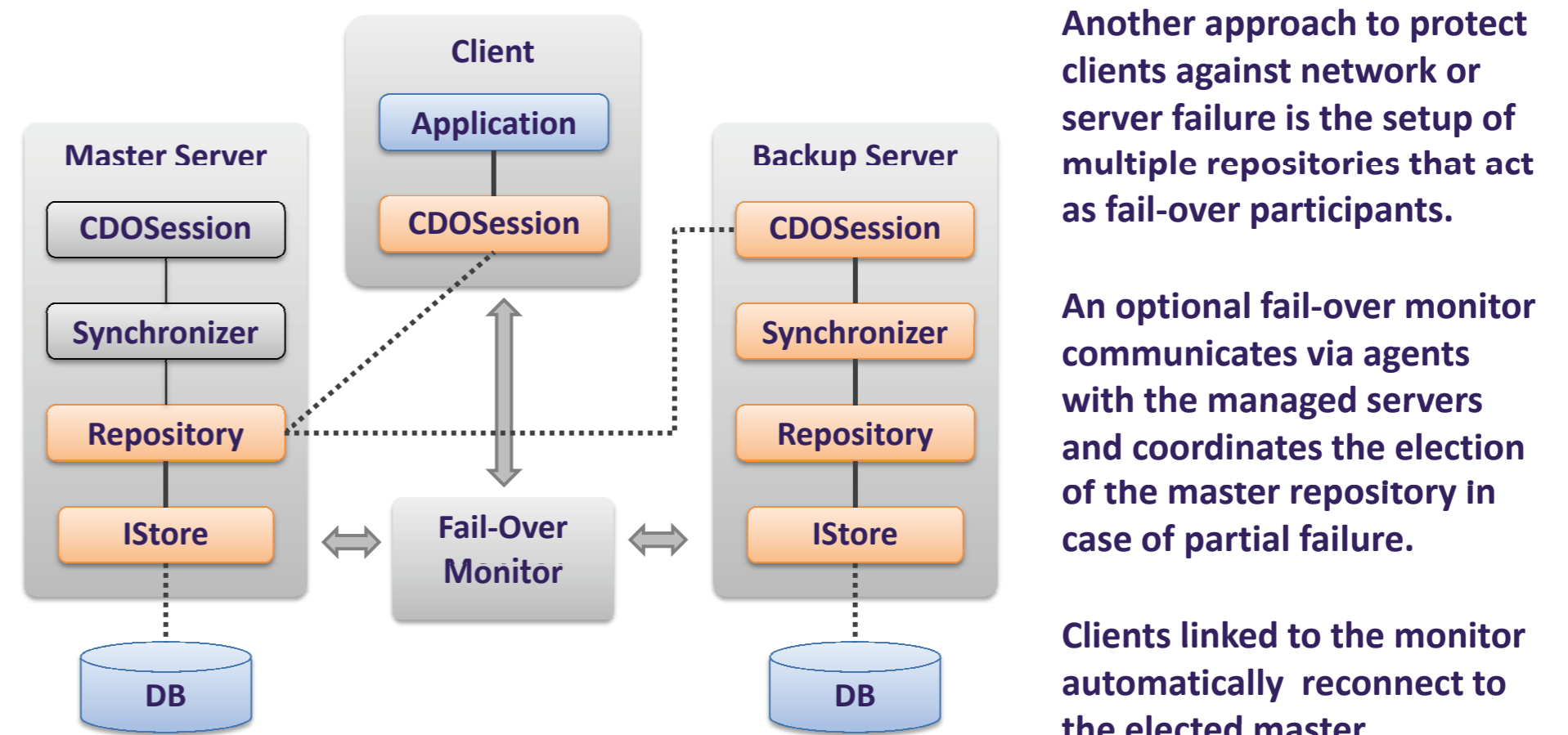
Offline Clone

Client
- Application
- CDOSession

Clone Server
- ISession
- OfflineClone — Synchronizer
- IStore — CDOSession
- DB

Master Server
- ISession
- MasterRepository
- IStore
- DB

Some usage scenarios require that applications stay fully functional even during periods of the master server being unavailable.

Cloned repositories continuously synchronize themselves in the background while they are online and offer read/write access all the time. Offline write access involves branching and merging.

Fail-Over

Master Server
- CDOSession
- Synchronizer
- Repository
- IStore
- DB

Client
- Application
- CDOSession

Backup Server
- CDOSession
- Synchronizer
- Repository
- IStore
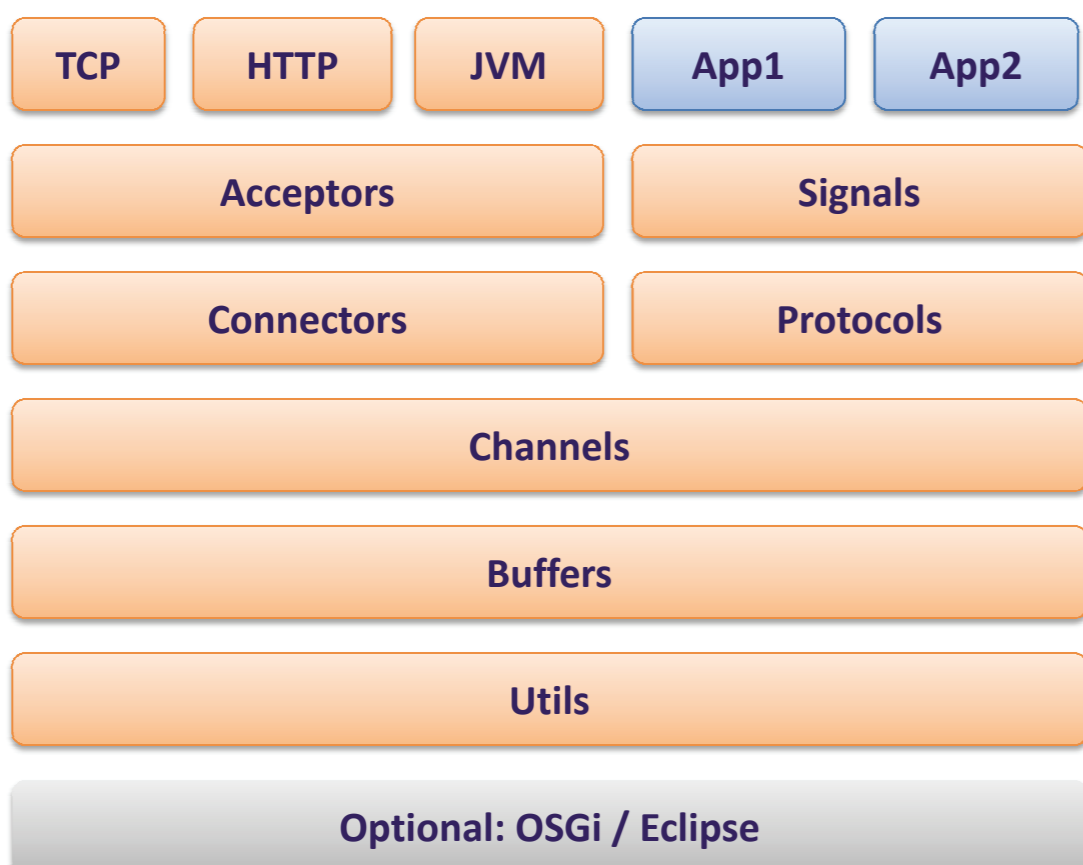- DB

Fail-Over Monitor

Another approach to protect clients against network or server failure is the setup of multiple repositories that act as fail-over participants.

An optional fail-over monitor communicates via agents with the managed servers and coordinates the election of the master repository In case of partial failure.

Clients linked to the monitor automatically reconnect to the elected master.

Net4j
The Signalling Platform

- TCP / HTTP / JVM / App1 / App2
- Acceptors / Signals
- Connectors / Protocols
- Channels
- Buffers
- Utils
- Optional: OSGi / Eclipse

High Performance:
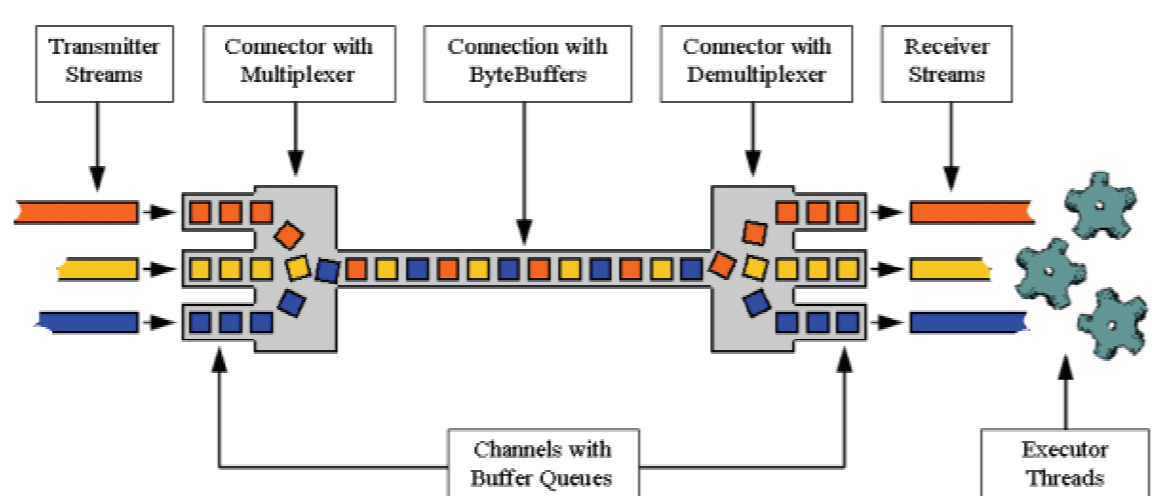java.nio.DirectByteBuffer, zero copying

Good Scalability:
java.nio.channels.Selector, single I/O thread possible

Multiple Transports:
Shipped with TCP, JVM and HTTP

Pluggable Protocols:
Independent of chosen transport

Server-initiated Push:
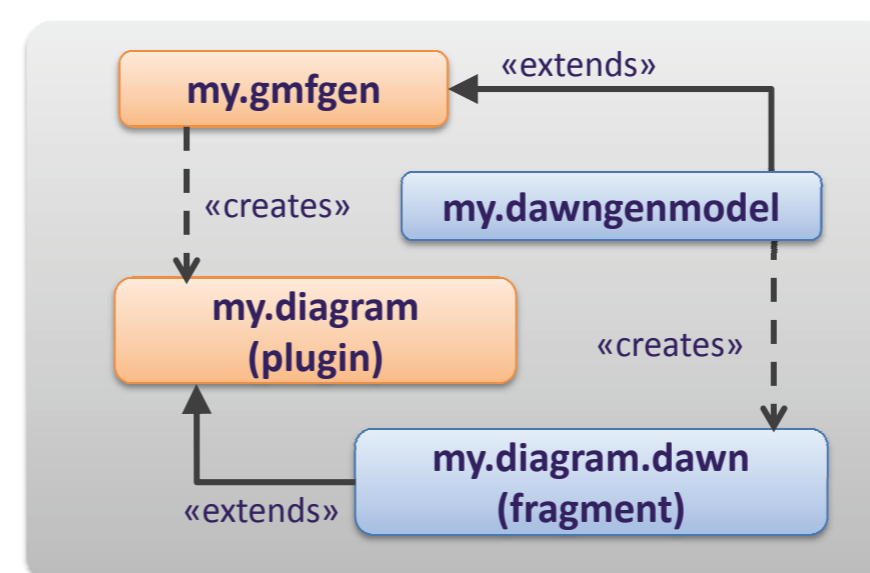Asynchronous and synchronous requests from the server to agents

Deployment Modes:
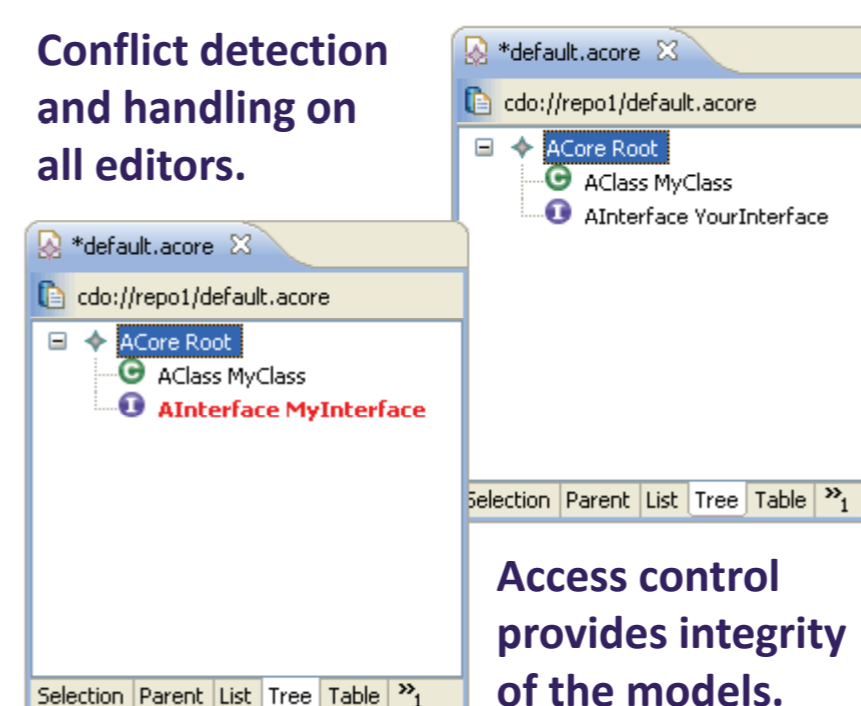Eclipse UI, pure OSGi, application servers or stand-alone

- Transmitter Streams
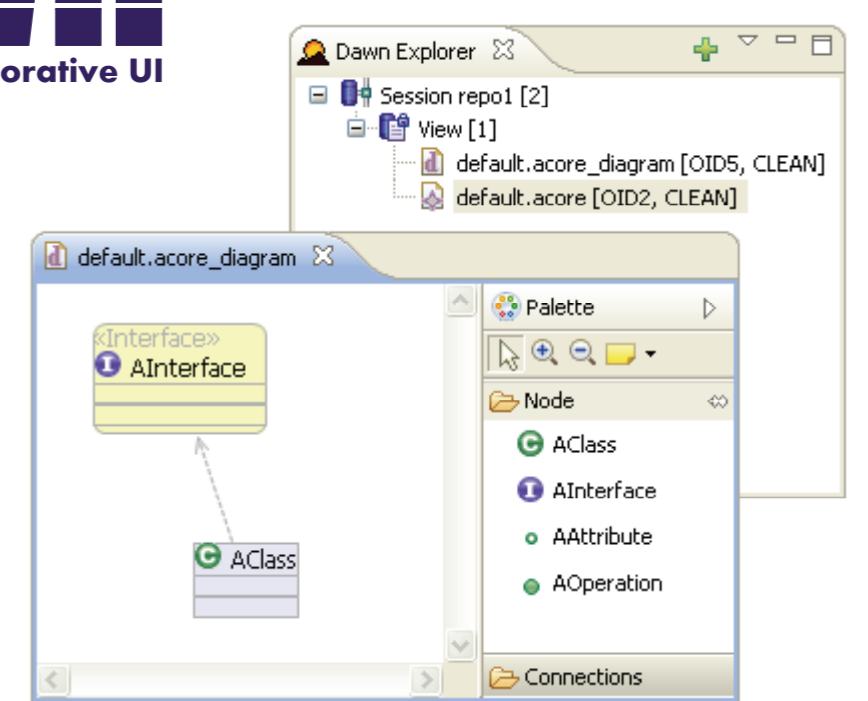- Connector with Multiplexer
- Connection with ByteBuffers
- Connector with Demultiplexer
- Receiver Streams
- Channels with Buffer Queues
- Executor Threads

Dawn
The Rise of the collaborative UI

Dawn provides collaboration based on CDO for existing user interfaces.

- my.gmfgen — «extends»
- «creates»
- my.dawngenmodel
- my.diagram (plugin) — «creates»
- «extends»
- my.diagram.dawn (fragment)

Generated extensions (fragments) for different kinds of EMF based editors (e.g. GMF, EMF, Graphiti).

Conflict detection and handling on all editors.

```
Dawn Explorer
▲ Session repo1 [2]
  ▲ View [1]
    default.acore_diagram [OID5, CLEAN]
    default..acore [OID2, CLEAN]
```

*default.acore
cdo:/repo1/default.acore
  ACore Root
    AClass MyClass
    AInterface YourInterface

*default.acore
cdo:/repo1/default.acore
  ACore Root
    AClass MyClass
    AInterface MyInterface

Selection Parent List Tree Table »

Access control provides integrity of the models.

Web-based solution for mobile devices and clients without installed Java environment.