



# Introduction To Cayenne

---

Bill Dudney

Virtuas Open Source Solutions





# R/O Mapping?

---

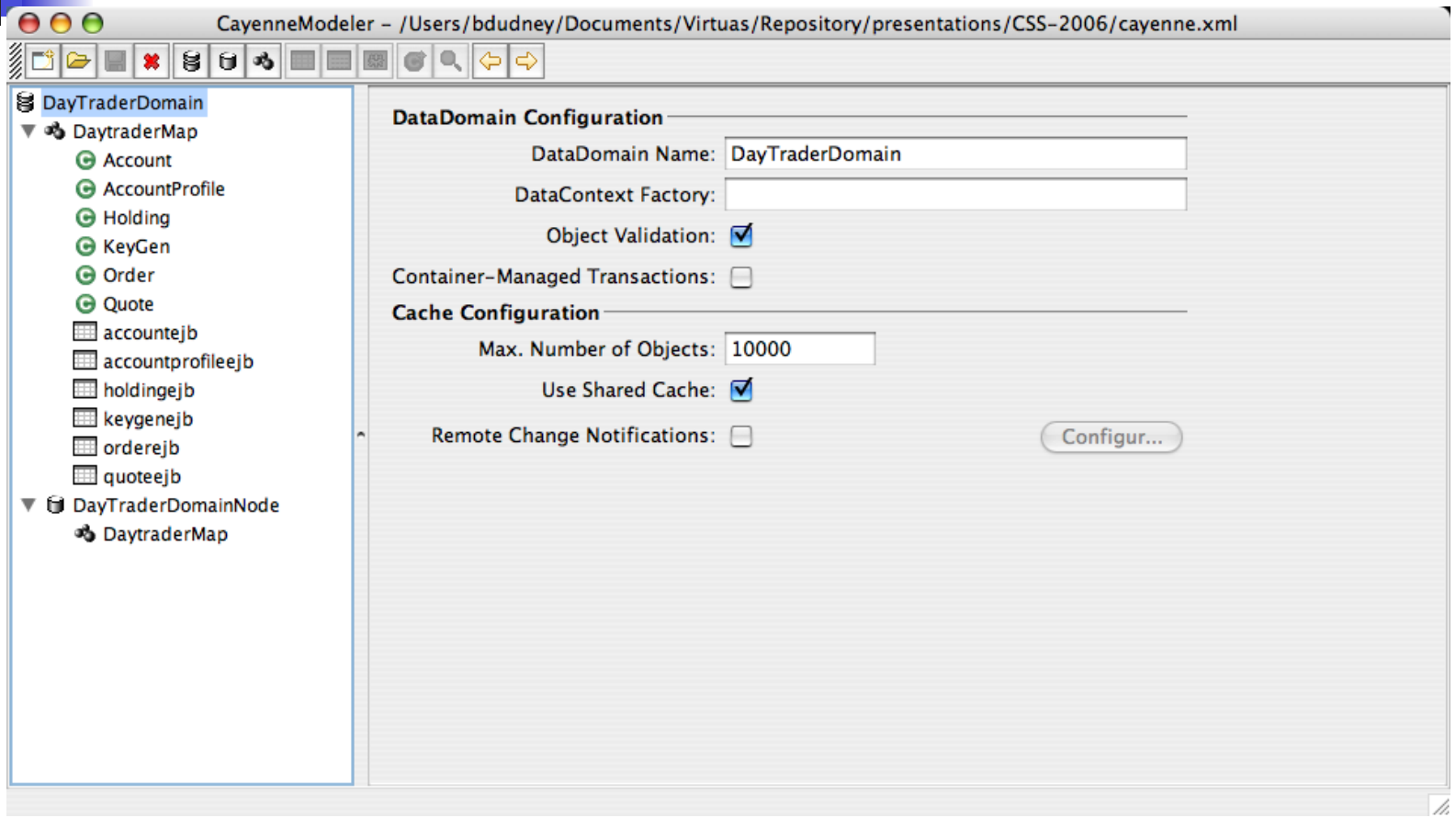
- Mainstream Technology
  - Hibernate
  - TopLink
  - EJB 3
- Long history of success (TopLink started in late 80's)



# The Successful MDA

---

# Cayenne Modeling



# Cayenne Modeling

The screenshot shows the CayenneModeler application window. The title bar reads "CayenneModeler - /Users/bdudney/Documents/Virtuas/Repository/presentations/CSS-2006/cayenne.xml". The interface is divided into a left-hand tree view and a right-hand configuration panel.

**Left Panel (Tree View):**

- DayTraderDomain
  - DaytraderMap
    - Account
    - AccountProfile
    - Holding
    - KeyGen
    - Order
    - Quote
    - accountejb
    - accountprofileejb
    - holdingejb
    - keygenejb
    - orderejb
    - quoteejb
  - DayTraderDomainNode
    - DaytraderMap

**Right Panel (DataMap Configuration):**

- DataMap Configuration**
  - DataMap Name: DaytraderMap
  - File: DaytraderMap.map.xml
  - DataNode: DayTraderDomainNode
- Entity Defaults**
  - DB Schema: [ ] Update...
  - Java Package: [ ] Update...
  - Custom Superclass: [ ] Update...
  - Optimistic Locking:  Update...
- Client Class Defaults**
  - Allow Client Entities:
  - Client Java Package: [ ] Update...



# Cayenne Modeling

The screenshot shows the CayenneModeler application window titled "CayenneModeler - /Users/bdudney/Documents/Virtuas/Repository/presentations/CSS-2006/cayenne.xml". The interface is divided into two main sections:

- Left Panel (Project Tree):** A tree view showing the project structure. The root is "DayTraderDomain", which contains a "DaytraderMap" and a "DayTraderDomainNode". Under "DaytraderMap", there are several entities: Account, AccountProfile, Holding, KeyGen, Order, and Quote. Under "DayTraderDomainNode", there are several entity jobs: accountejb, accountprofileejb, holdingejb, keygenejb, orderejb, and quoteejb. The "DayTraderDomainNode" and its "DaytraderMap" child are currently selected.
- Right Panel (Configuration):** This panel has two tabs: "Main" (selected) and "Adapter". It contains two main configuration sections:
  - DataNode Configuration:**
    - DataNode Name: DayTraderDomainNode
    - Local DataSource (opt.): Select DataSource for Local Work... (with a dropdown arrow and a "..." button)
    - DataSource Factory: org.objectstyle.cayenne.conf.DriverDataSourceFactory (with a dropdown arrow)
  - JDBC Configuration:**
    - JDBC Driver: com.mysql.jdbc.Driver
    - DB URL: jdbc:mysql://localhost:3306/daytrader
    - User Name: daytrader
    - Password: \*\*\*\*\*
    - Min Connections: 5
    - Max Connections: 15
    - A "Sync with Local" button is located at the bottom right of this section.



# Cayenne Modeling

The screenshot shows the CayenneModeler application window titled 'CayenneModeler - /Users/bdudney/Documents/Virtuas/Repository/presentations/CSS-2006/cayenne.xml'. The interface is divided into a left-hand project tree and a main central pane. The central pane is currently displaying the 'Attributes' tab for an entity. The project tree on the left shows a hierarchy starting with 'DayTraderDomain', followed by 'DayTraderMap', and then several entities including 'Account', 'AccountProfile', 'Holding', 'KeyGen', 'Order', 'Quote', and several 'ejb' entities. The main pane shows a table with the following columns: 'ObjAttribute', 'Java Type', 'DbAttribute', 'DB Type', and 'Used for Lockin'. The table lists attributes for the 'Account' entity, such as 'balance', 'creationDate', 'lastLogin', 'loginCount', 'logoutCount', 'openBalance', and 'profileUserID', each with its corresponding Java and database types and a checked box indicating it is used for locking.

| ObjAttribute  | Java Type            | DbAttribute    | DB Type   | Used for Lockin                     |
|---------------|----------------------|----------------|-----------|-------------------------------------|
| balance       | java.math.BigDecimal | balance        | DECIMAL   | <input checked="" type="checkbox"/> |
| creationDate  | java.util.Date       | creationdate   | TIMESTAMP | <input checked="" type="checkbox"/> |
| lastLogin     | java.util.Date       | lastlogin      | TIMESTAMP | <input checked="" type="checkbox"/> |
| loginCount    | java.lang.Integer    | logincount     | INTEGER   | <input checked="" type="checkbox"/> |
| logoutCount   | java.lang.Integer    | logoutcount    | INTEGER   | <input checked="" type="checkbox"/> |
| openBalance   | java.math.BigDecimal | openbalance    | DECIMAL   | <input checked="" type="checkbox"/> |
| profileUserID | java.lang.String     | PROFILE_USERID | VARCHAR   | <input checked="" type="checkbox"/> |



# Cayenne Modeling

The screenshot shows the CayenneModeler application window. The title bar reads "CayenneModeler - /Users/bdudney/Documents/Virtuas/Repository/presentations/CSS-2006/cayenne.xml". The interface is divided into a left-hand tree view and a main central pane.

The left-hand tree view shows a domain structure:

- DayTraderDomain
  - DaytraderMap
    - Account
    - AccountProfile
    - Holding
    - KeyGen
    - Order
    - Quote
    - accountejb (selected)
    - accountprofileejb
    - holdingejb
    - keygenejb
    - orderejb
    - quoteejb
  - DayTraderDomainNode
    - DaytraderMap

The main pane is titled "Attributes" and contains a table with the following data:

| Name           | Type      | PK                                  | Mandatory                           | Max Length | Precision |
|----------------|-----------|-------------------------------------|-------------------------------------|------------|-----------|
| PROFILE_USERID | VARCHAR   | <input type="checkbox"/>            | <input type="checkbox"/>            | 250        |           |
| accountid      | INTEGER   | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 11         |           |
| balance        | DECIMAL   | <input type="checkbox"/>            | <input type="checkbox"/>            | 10         | 2         |
| creationdate   | TIMESTAMP | <input type="checkbox"/>            | <input type="checkbox"/>            | 19         |           |
| lastlogin      | TIMESTAMP | <input type="checkbox"/>            | <input type="checkbox"/>            | 19         |           |
| logincount     | INTEGER   | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | 11         |           |
| logoutcount    | INTEGER   | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | 11         |           |
| openbalance    | DECIMAL   | <input type="checkbox"/>            | <input type="checkbox"/>            | 10         | 2         |







# Demo

---





# Getting At Data

---

## The Context API



# Running Queries

---

```
public java.util.List performQuery(Query query)
```



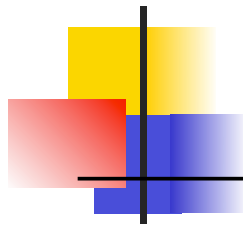
public void commitChanges()



```
public void deleteObject(java.lang.Object object)
```



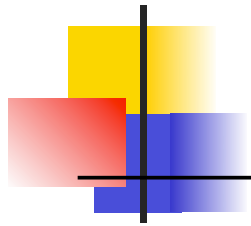
```
public java.lang.Object newObject(java.lang.Class clazz)
public void registerNewObject(java.lang.Object object)
```



# Code

---





# Remotely Getting At Data

---





# API Stays the Same

---

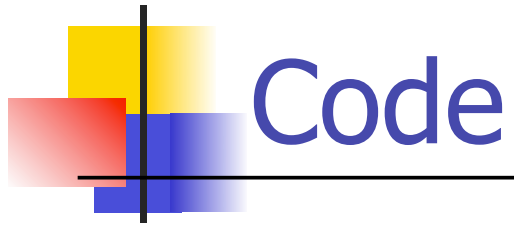
```
public java.util.List performQuery(Query query)
```

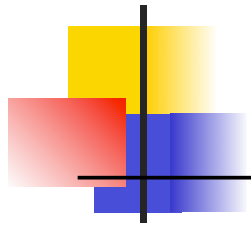
```
public void commitChanges()
```

```
public void deleteObject(java.lang.Object object)
```

```
public void registerNewObject(java.lang.Object object)
```

```
public java.lang.Object newObject(java.lang.Class clazz)
```





# Conclusion & Questions

---

