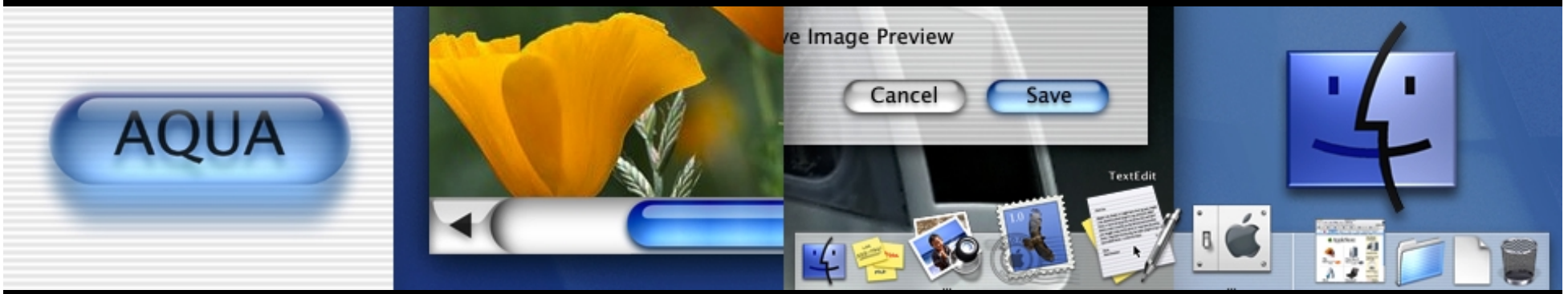
Session 415

WebObjects: Advanced EOF



Eric Noyau

Introduction

- Explain the fine points of EOF and how you can take advantage of them
- Review the common misunderstanding of the product
- Give you easy to follow guidelines to increase your chances of success

What You'll Learn

- How to design a good Enterprise Object (EO)
- New 4.5 features you cannot do without

How to Design a Good EO

- Build your data model or make sure you understand it well

How to Design a Good EO

Data Model

- Review the performance implications of your design
  - Try to move BLOBS in their own tables
  - Be careful with inheritance
- Triggers and stored procedures in your database are making your job harder
- Think about how your primary keys are generated

How to Design a Good EO

Data Model: Primary key generation

- If you can, let the EOAdaptor handle it (sequences, stored procedure or separate table)
- Make the primary key a 12 Byte binary
- Use the delegate: **databaseContextNewPrimaryKey(dbContext, object, entity)**
- Mark the primary key attributes class property and fill them yourself in **awakeFromInsertion()**

How to Design a Good EO

- Build your data model or make sure you understand it well
- Build your EOModel file and validate it

How to Design a Good EO

EOModel file tips

- Spend time on your EOModel file
- Click on all the inspectors, make sure you understand all the checkboxes
- Compare your relationships with the ones in the example EOModels
- If possible, build a Direct to Java Client application

How to Design a Good EO

- Build your data model or make sure you understand it well
- Build your EOModel file and validate it
- Implement your logic

How to Design a Good EO

Implement your logic

- EOGenericRecord
  - No code
  - No logic, just the data
- Custom Enterprise Object
  - More powerful, full-fledged objects
  - Maintenance when the model changes

How to Design a Good EO

Implement your logic

- Subclass EOGenericRecord: less maintenance and improved Java performance


```
public class Foo extends EOGenericRecord {
  public final String Name = “name”;
  public String name() {
      return (String)storedValueForKey(Name);
  }
  public void setName(String name) {
      takeStoredValueForKey(name, Name);
  }
}
```

How to Design a Good EO

Implement your logic

	Get	Set
Key Value Coding	String name() String name String _name() String _name	void setName(String name) String name void _setName(String name) String _name
Stored Key Value Coding	String _name() String _name String name() String name	void _setName(String name) String _name void setName(String name) String name

public static boolean canAccessFieldsDirectly()
public static boolean shouldUseStoredAccessors()

How to Design a Good EO

Implement your logic

- Validate your attributes

```
public Object validateName(Object value)
        throws EOValidation.Exception
```

- Validate your EO

```
public void validateForSave()
public void validateForDelete()
public void validateForInsert()
        throws EOValidation.Exception
```

  - Call super()! No update!
  - Called only at the end of the event or request/response loop

How to Design a Good EO

Implement your logic

- Methods you can implement:
  public void unableToSetNullForKey()
  public void awakeFromInsertion()
  public void awakeFromFetch()
- Methods you should not override:
  public void willChange()
  public void willRead()

How to Design a Good EO

Implement your logic

- Relationship management

addObjectToBothSideOfRelationshipWithKey()

How to Design a Good EO

Keep your EOs clean

- Do not put EOAccess specific code in your EO
- Do not put WOFramework code in your EO

How to Design a Good EO

- Build your data model, or make sure you understand it well
- Build your EOModel file and validate it
- Implement your logic

4.5 Features

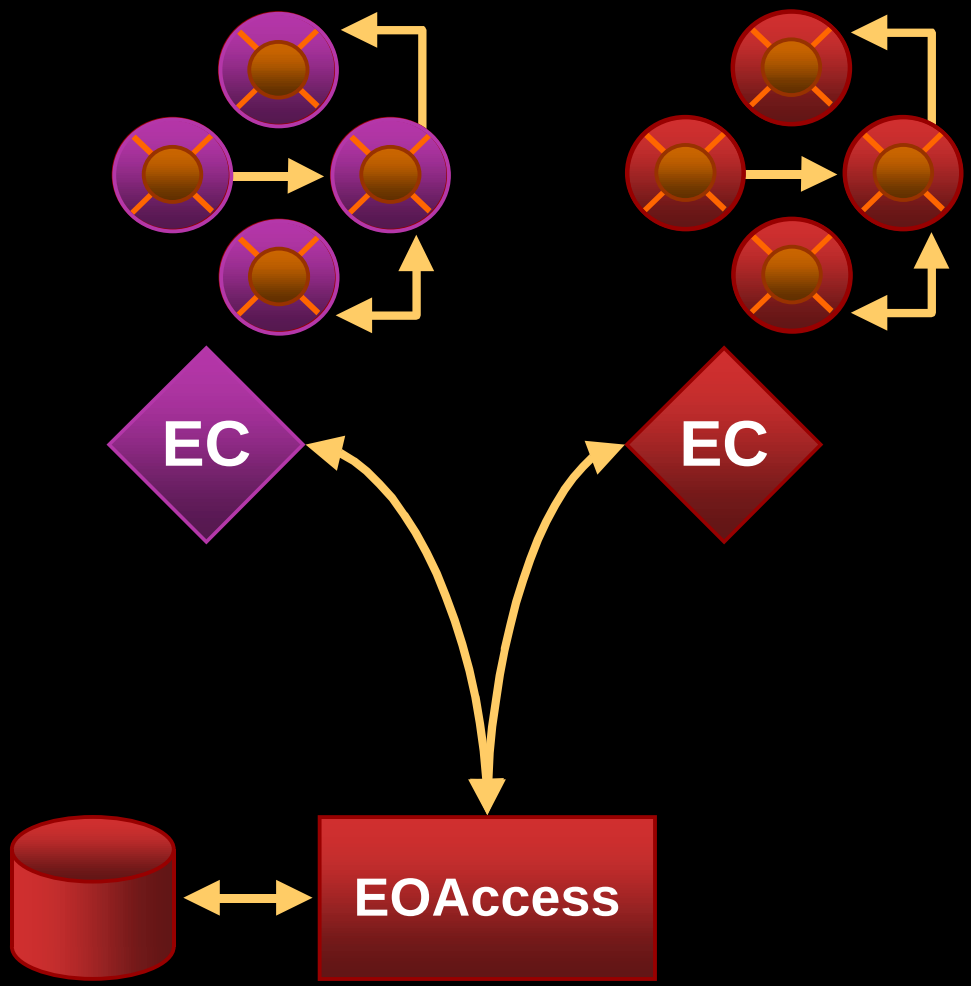
4.5 Features

Subclassing EOGenericRecord

- Performance
- Less maintenance, more flexible

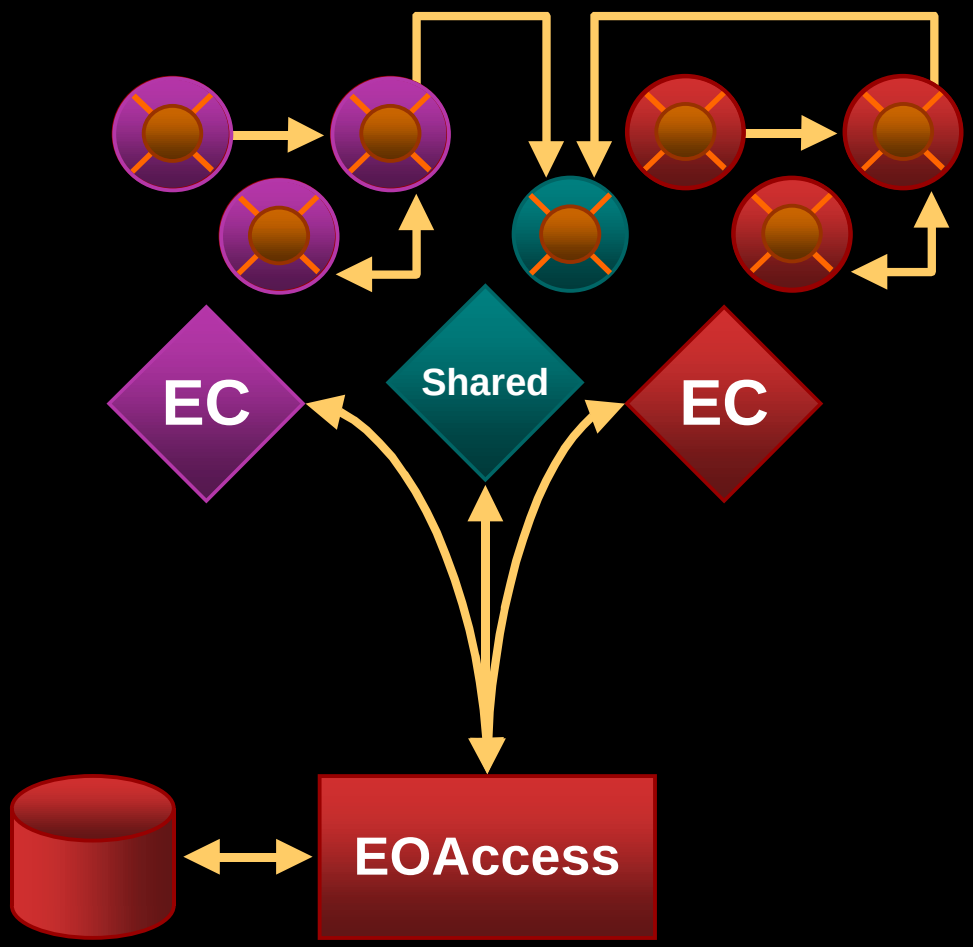
4.5 Features

Shared Editing Context



4.5 Features

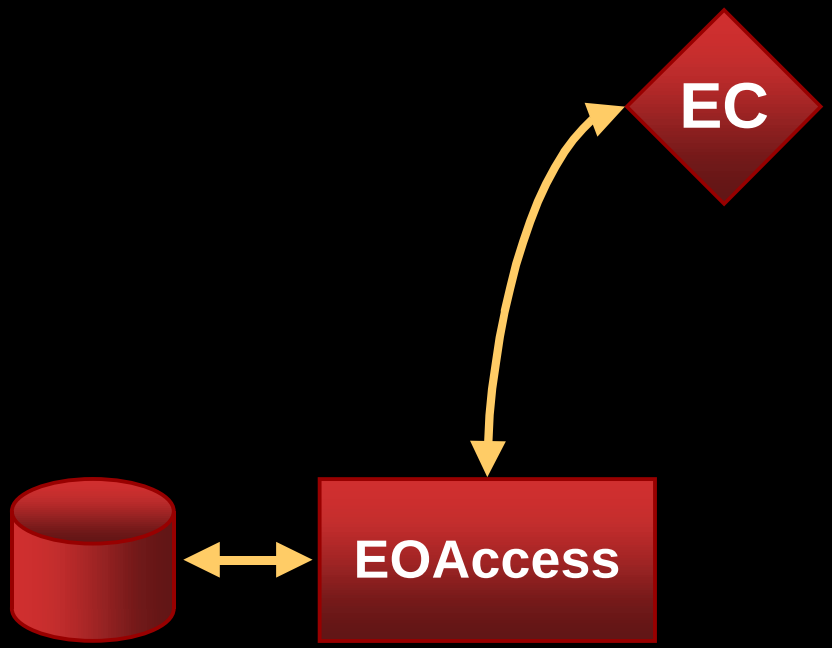
Shared Editing Context



- Read only EditingContext
- Relationship destination only
- Uniquing scope changed

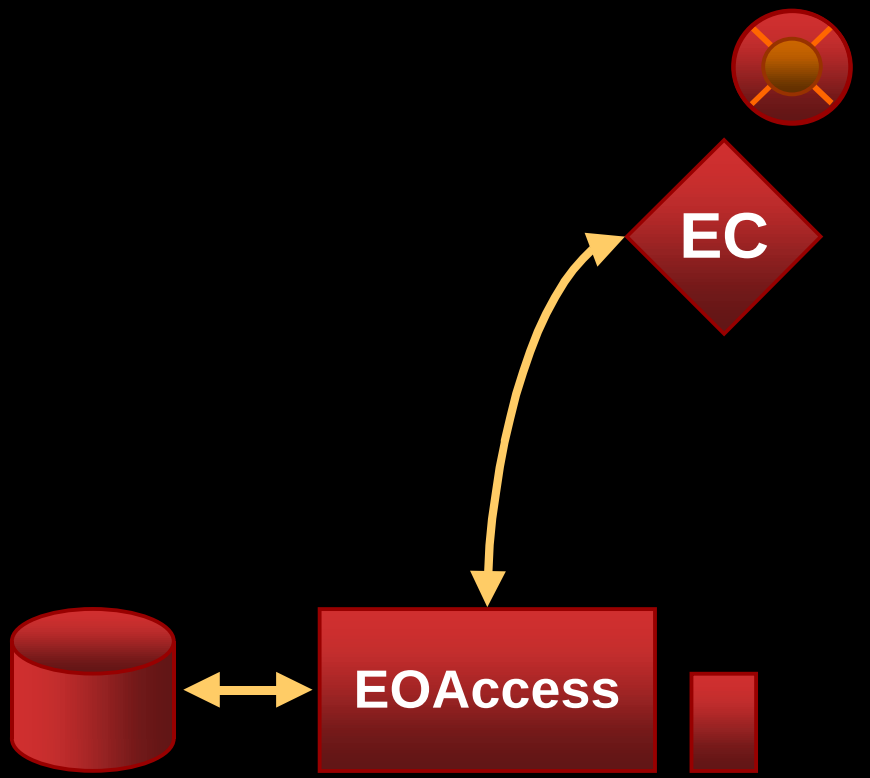
4.5 Features

Snapshot Refcounting



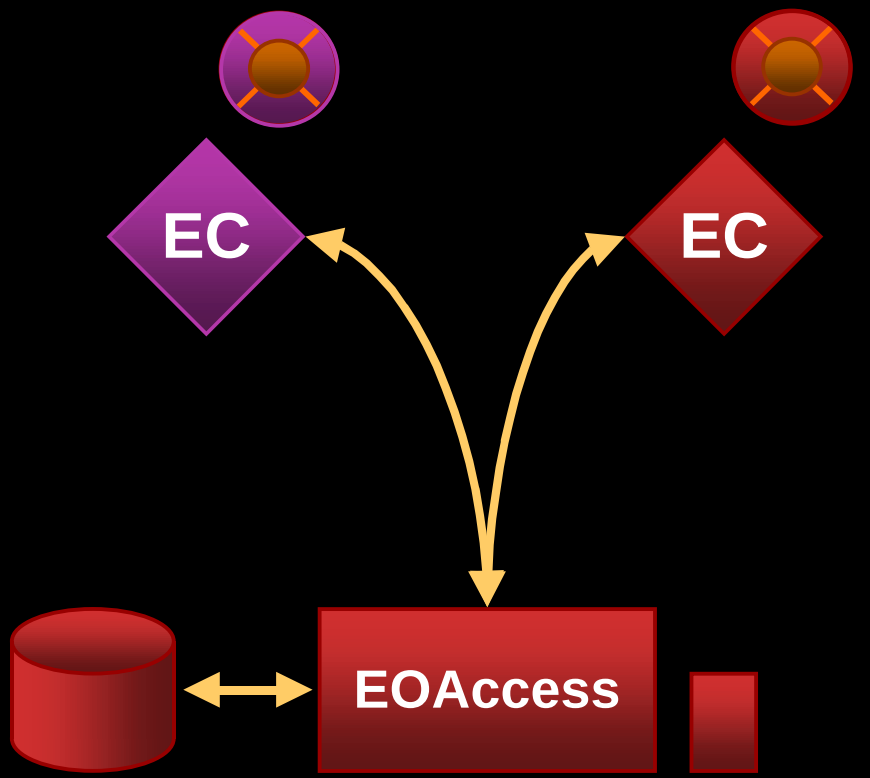
4.5 Features

Snapshot Refcounting



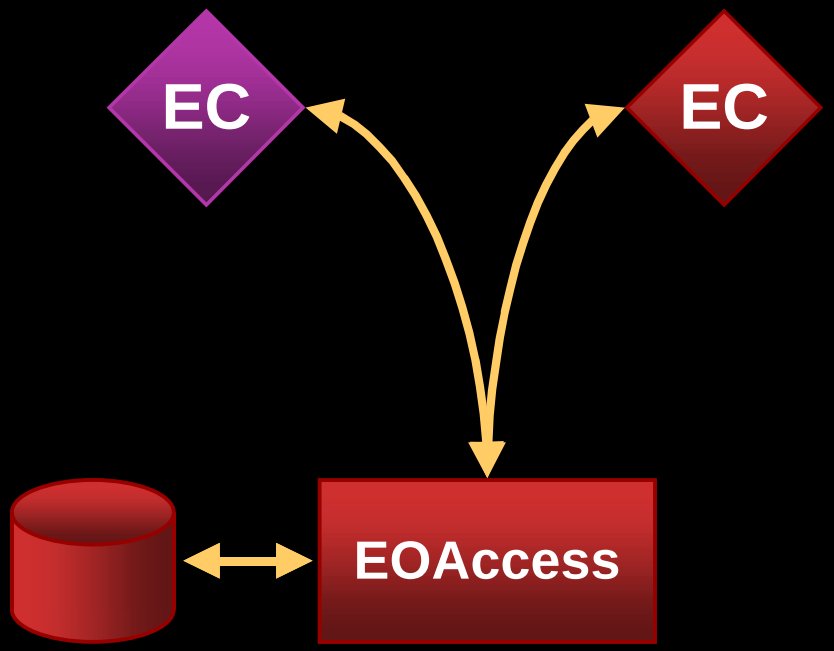
4.5 Features

Snapshot Refcounting



4.5 Features

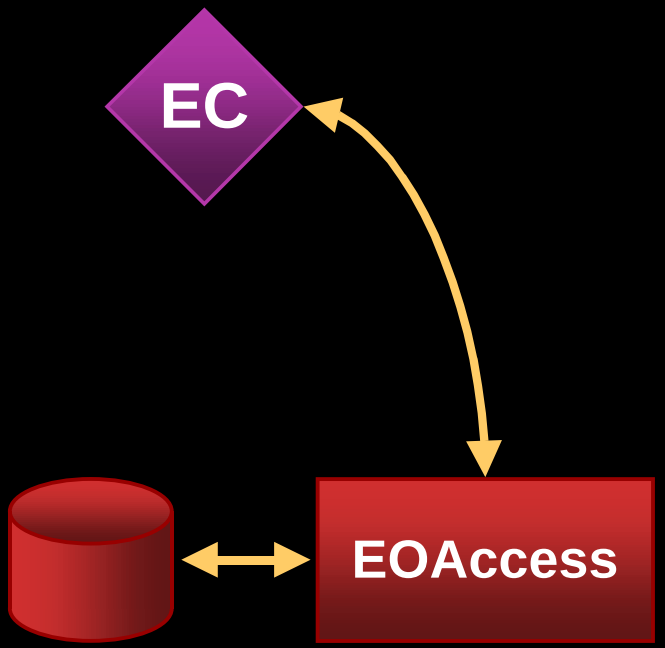
Snapshot Refcounting



- Reduce memory footprint
- Help snapshot freshness

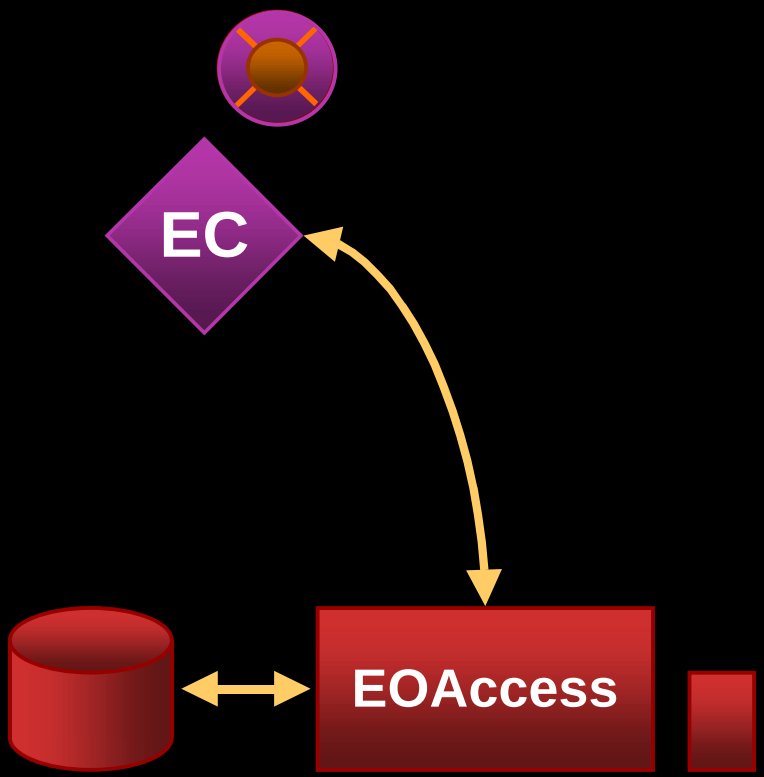
4.5 Features

Snapshot Timestamp



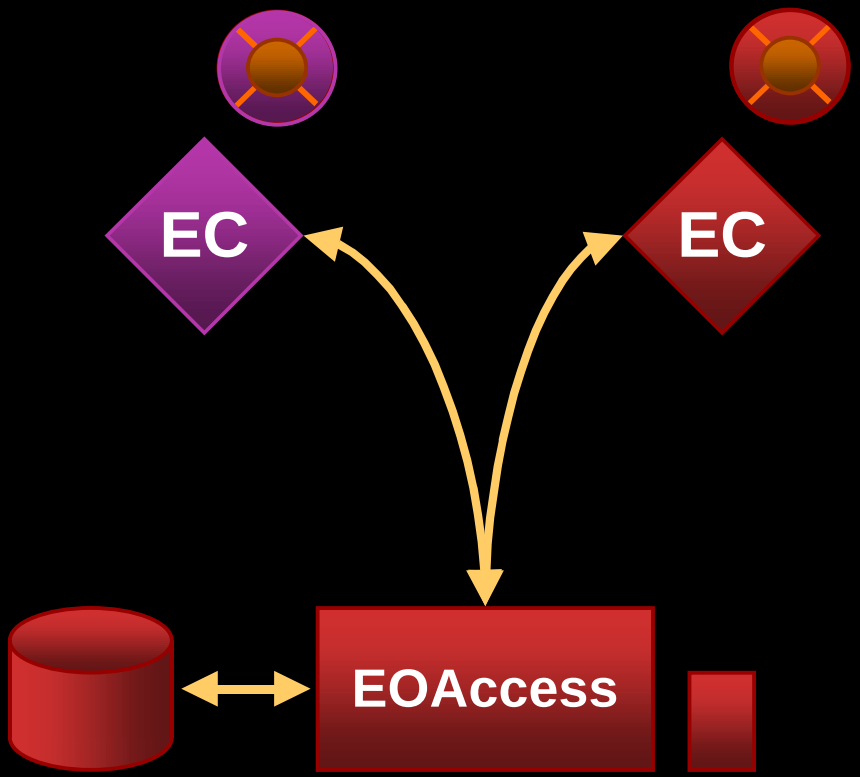
4.5 Features

Snapshot Timestamp



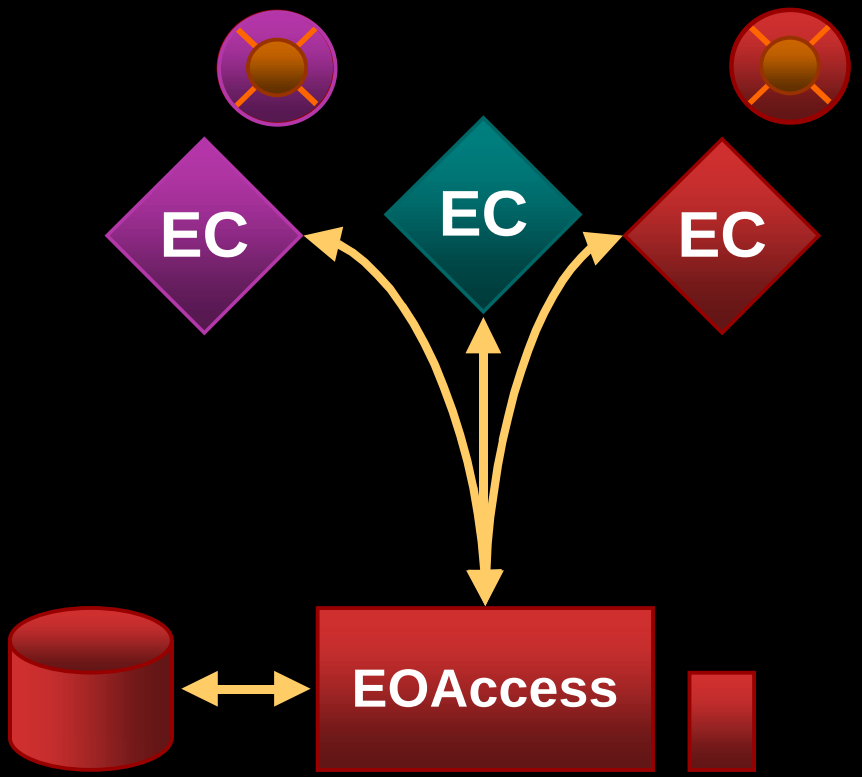
4.5 Features

Snapshot Timestamp



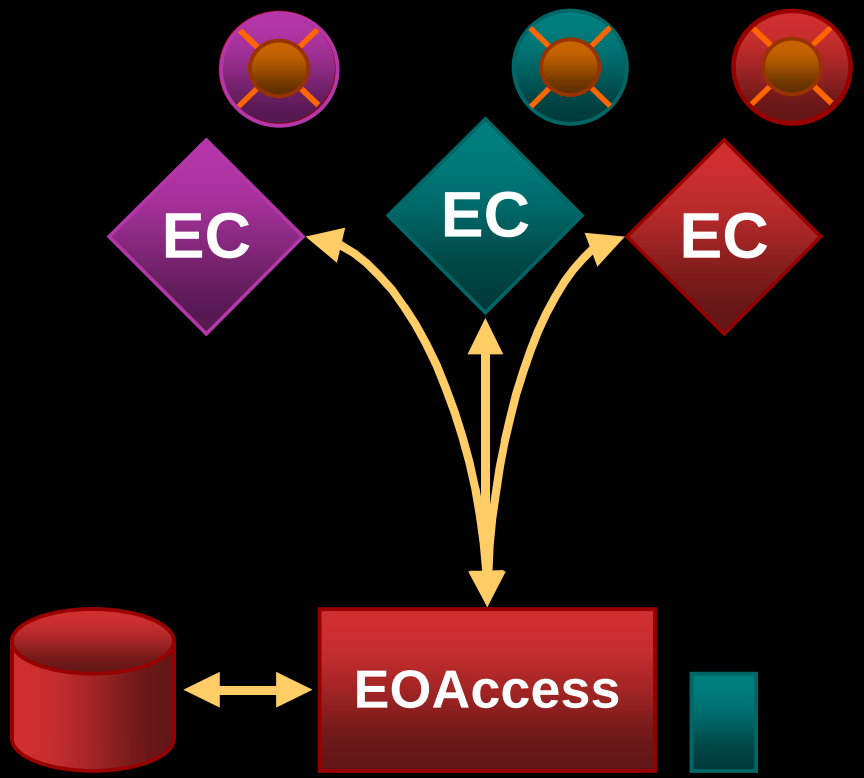
4.5 Features

Snapshot Timestamp



4.5 Features

Snapshot Timestamp

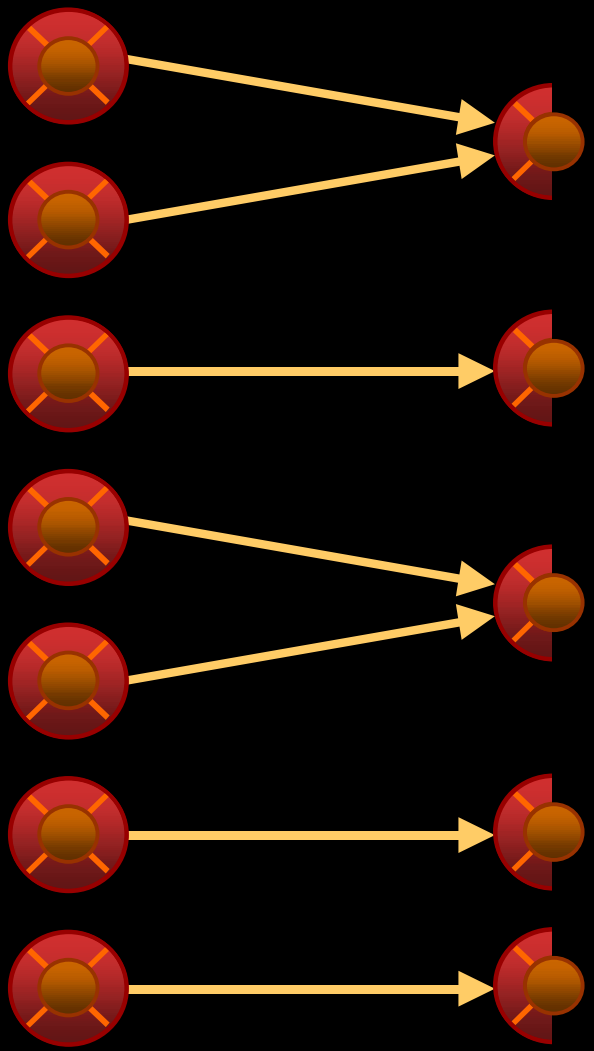


- No more stale data
- You control the freshness of the data
- Easy way to get updated data

setDefaultFetchTimestampLag()
setFetchTimeStamp()

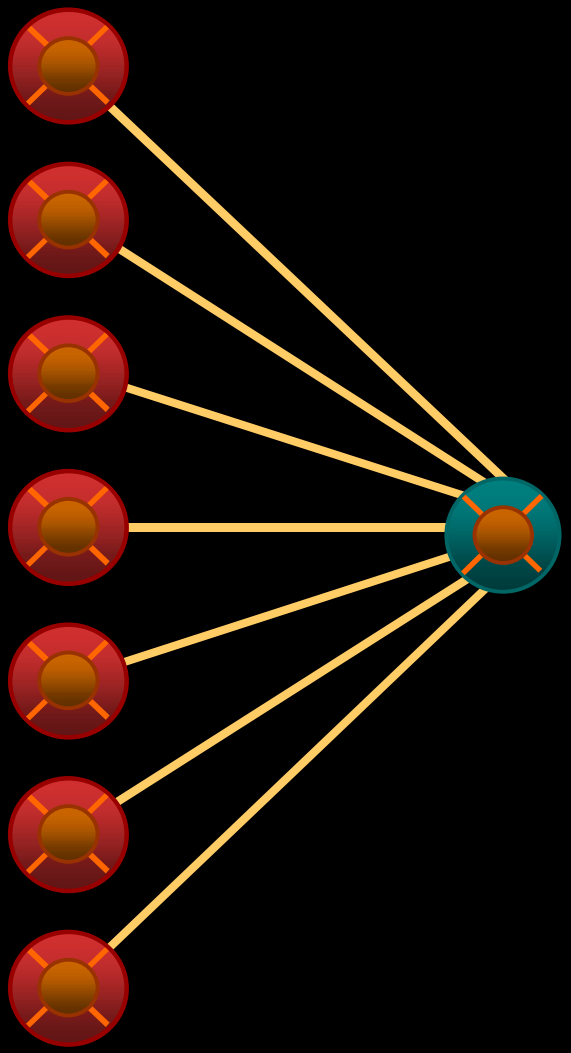
4.5 Features

Deferred Faulting



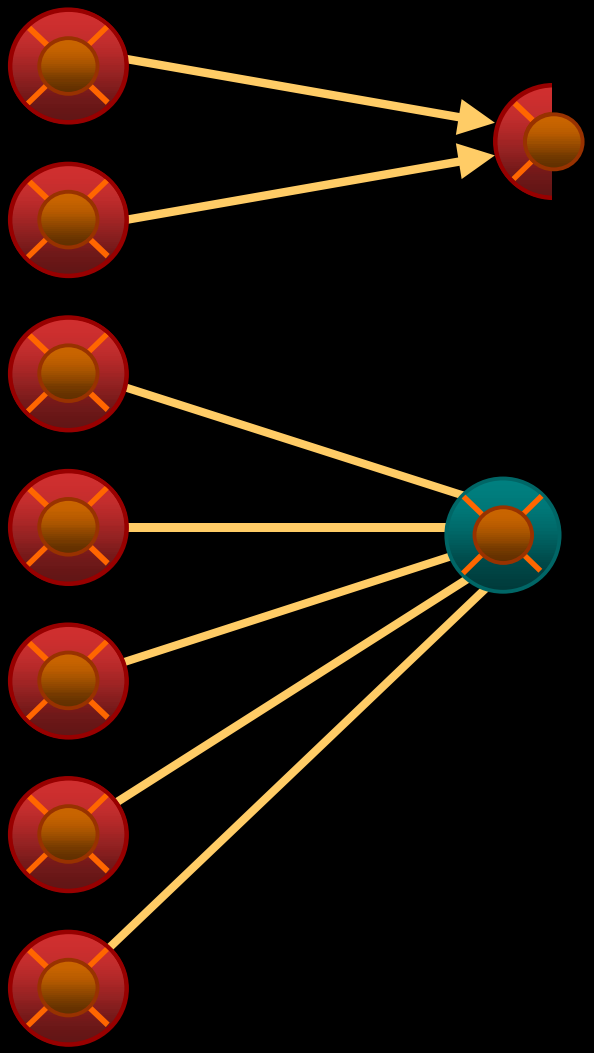
4.5 Features

Deferred Faulting



4.5 Features

Deferred Faulting



- Optional
- Faster fetches

willReadRelationship()

4.5 Features

Database reconnection

- Automatic on EOAdaptor exception
- Modify the behavior by using the delegates

reconnectionDictionaryForAdaptor()
databaseContextShouldHandleDatabaseException()

4.5 Features

Handling of missing faults

- No exception on fetch
- Empty EO inserted
- Exception thrown on save
- Use the delegates to control the behavior

  databaseContextFailedToFetchObject()

4.5 Features

Delegates made easy

- On EOAdaptor, EOAdaptorContext, and EODatabaseContext

  setDefaultDelegate()

- Do it when you start the application and be done with it

4.5 Features

LDAP Adaptor

- You asked for it…
- No EOModel needed for authentication:

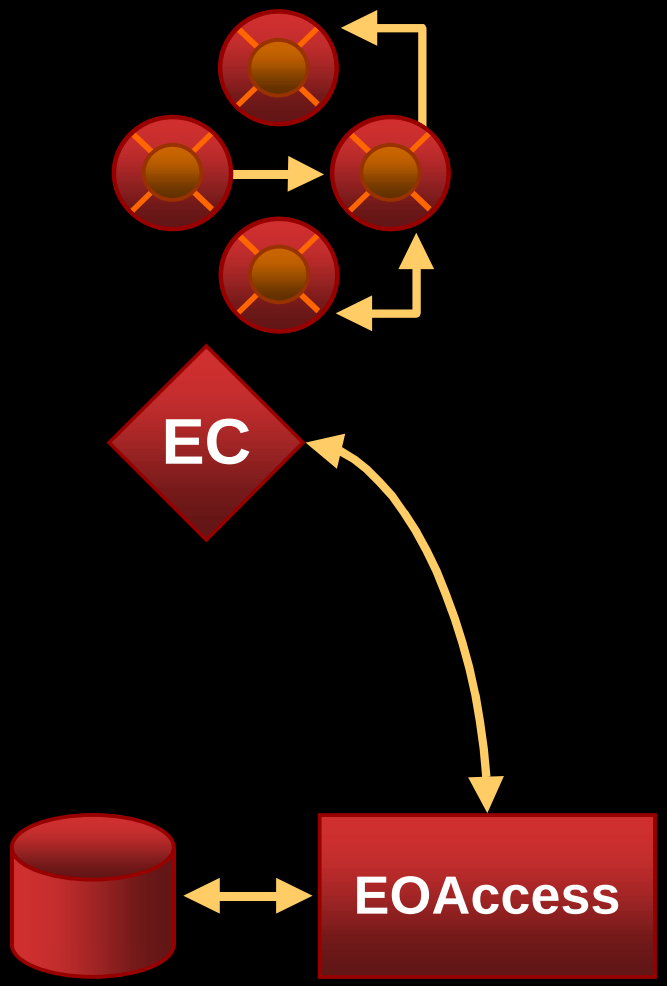
```
LDAPAdaptor.authenticateUser (
        userName, password,
        "bigbird.apple.com",
        "cn", "o=Apple Computer", "Subtree");
```

4.5 Features

- Subclassing EOGenericRecord
- Shared editing context
- Snapshot refcounting/timestamp
- Deferred faulting
- DB reconnect
- Handling of missing faults
- Easy delegates
- LDAP Adaptor

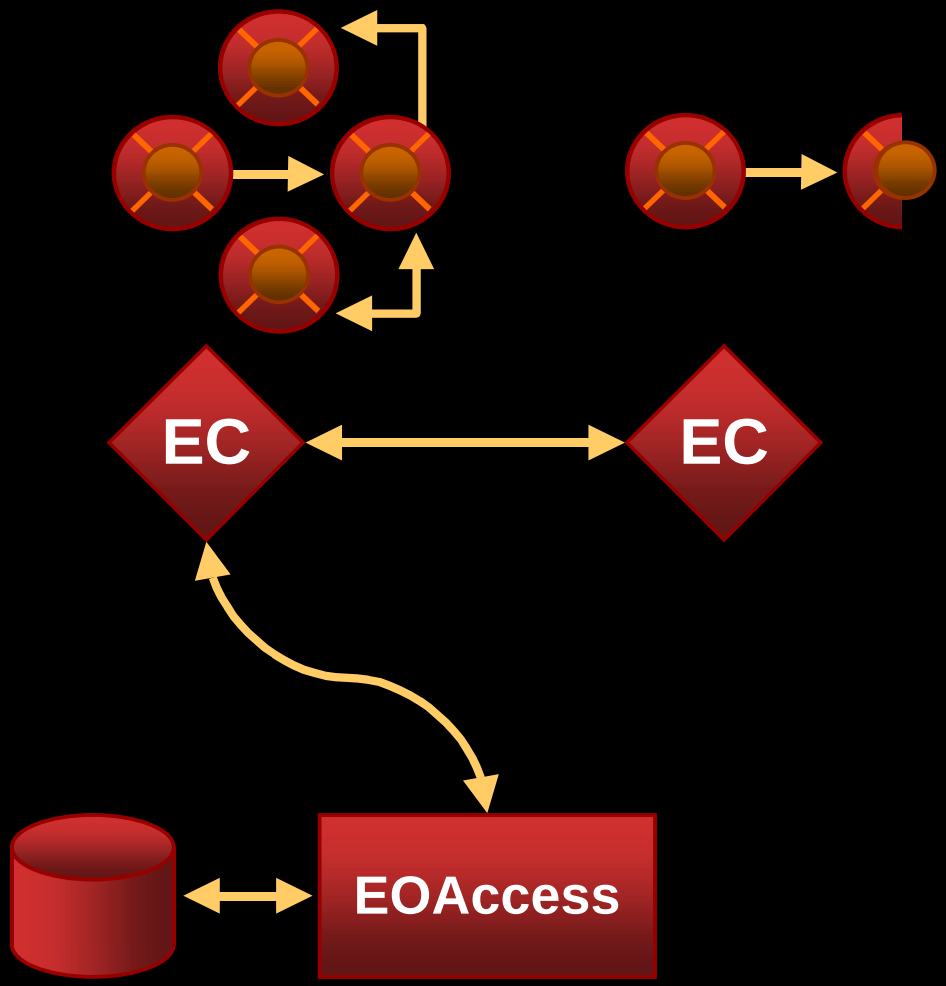
Editing Context

Nested Editing Context



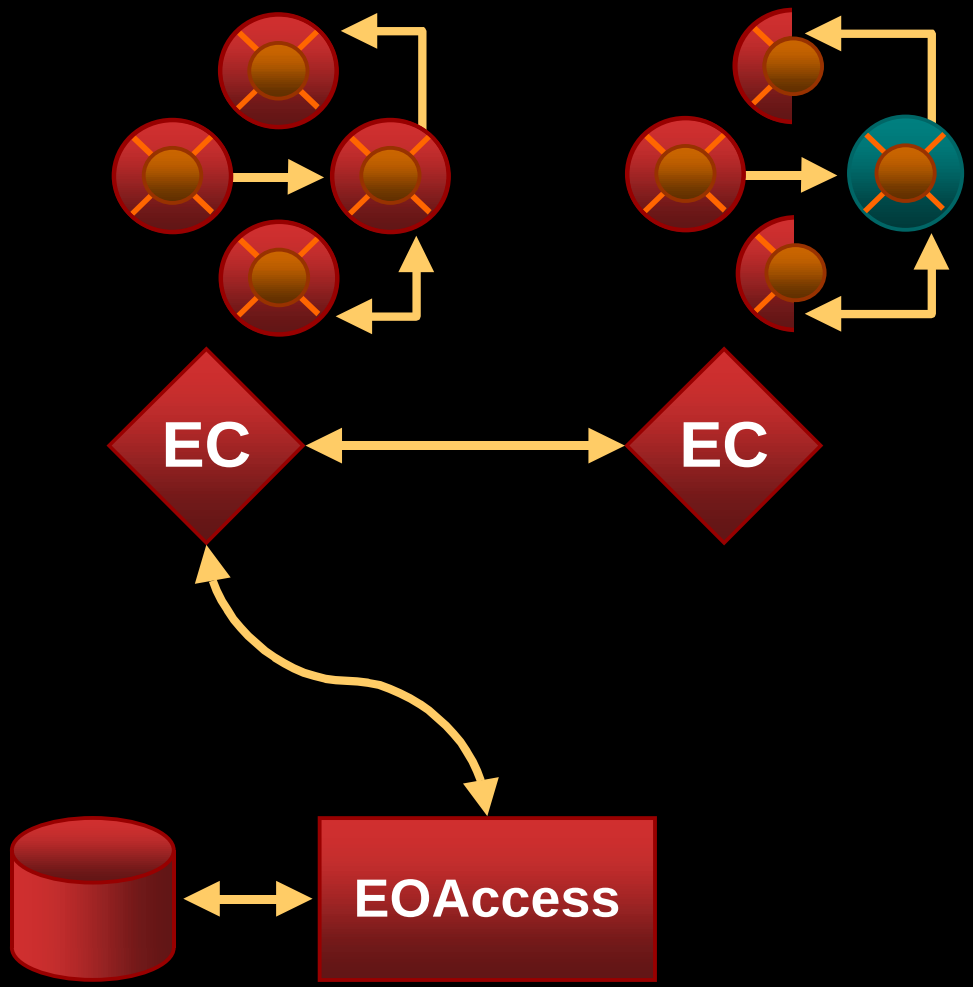
Editing Context

Nested Editing Context



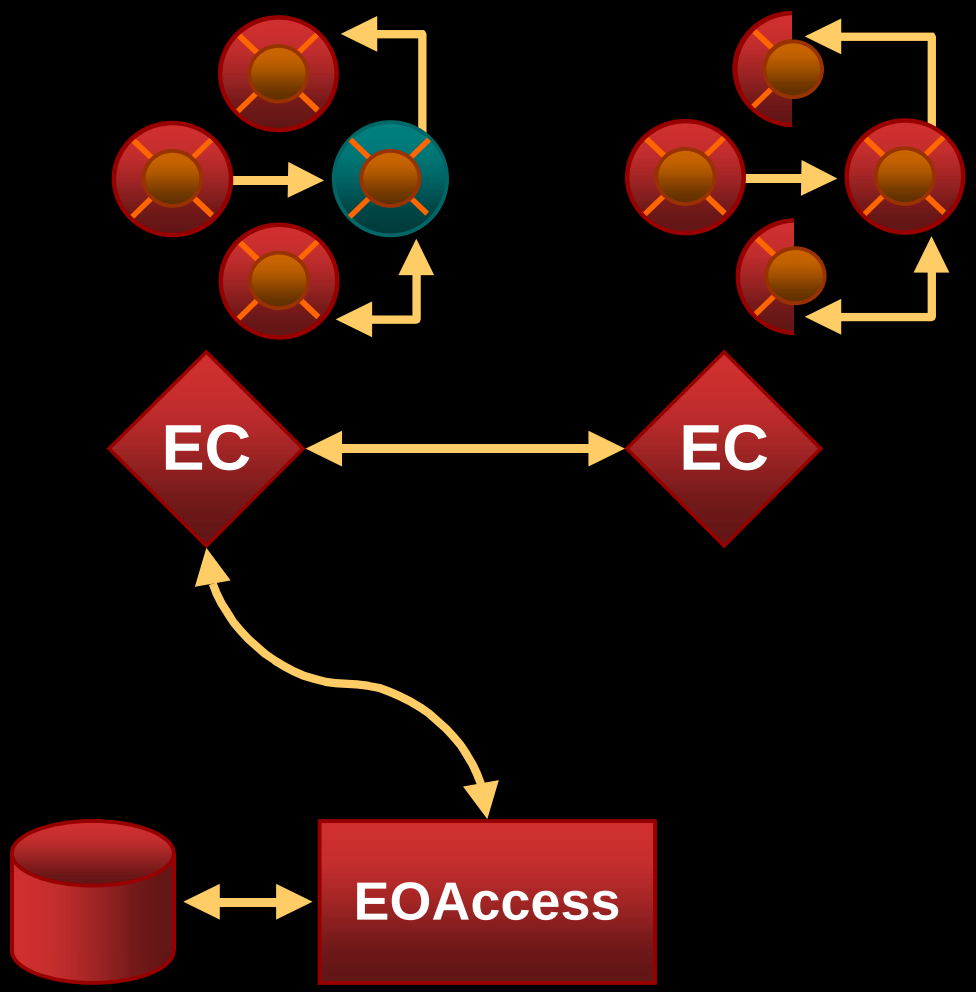
Editing Context

Nested Editing Context



Editing Context

Nested Editing Context



- For changes you may not want to save
- Avoid data pollution on browser backtrack
- Creating an EditingContext is cheap

What's Happening at the End of an Event?

- "event"? What's an event?
- Delete propagation and validation
- Undo registration/bracketing

The End

The End

Work with EOF, not against it

Eric Noyau
Josh Fagans

For More Information

http://www.apple.com/webobjects

http://enterprise.apple.com/wwdc2000

Visit the WebObjects lab downstairs!
Everyday from 11:00 a.m.–2:00 p.m.

Try out your WebObjects 4.5 Evaluation CD!

Roadmap

915 Feedback Forum: WebObjects

Room J2
Fri., 3:30 p.m.

Roadmap

403 WebObjects: EOModeler Mapping Data To Objects	Room J2 **Tue., 3:00 p.m.**
406 WebObjects: Direct To Java Client Code free desktop applications	Room J2 **Wed., 10:30 a.m.**
412 WebObjects: EOF Caching Caching and synchronization strategies	Room J2 **Thurs., 2:00 p.m.**

Who to Contact

Toni Trujillo Vian
Director, WebObjects Engineering
wofeedback@group.apple.com

Ernest Prabhakar
Product Line Manager, WebObjects
webobjects@group.apple.com

WWDC

Worldwide Developers Conference 2000

Think different.