PRAIRIE DEV CON 2010

PRAIRIE DEVELOPER CONFERENCE

Microsoft

adxstudio

imaginet SOLVERA paradigm

Developer Foundation Track

Sponsored By

JetBRAINS

ORM

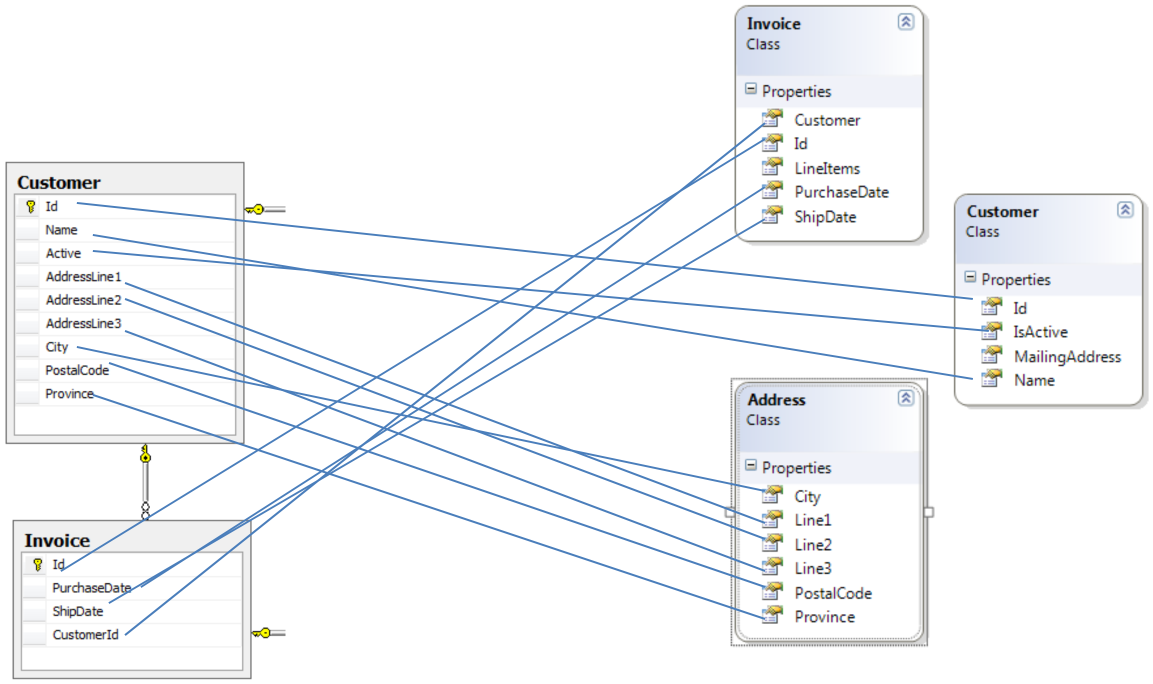
Fundamentals of Object Relational Mappers

DATA ACCESS…WE ALL DO IT

- Very few applications without DALs
- We build them over and over
- They change slightly, but still solve the same problems
- Rarely are they feature complete

Impedance Mismatch

- RDBMS to OO mapping doesn't work well
- Your Objects shouldn't match your data structure
- Converting from DB into Objects is what DALs do
- Difficult to properly and completely solve



Basic Requirements Of a DAL

- ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- Transactional
- Connection Style Agnostic
- Data Programming Style Agnostic

More Requirements Of A DAL

- Unit Of Work
- Caching
- Lazy Loading
- Persistence Ignorance
- Persistence By Reachability
- Versioning/Deployment Story

Stop Writing It Over And Over

- Frameworks
  - nHibernate
  - iBatis
  - Linq2Sql
  - Entity Framework
  - LLBLGen, CSLA, etc.
- Code Generation

The Code Gen Situation

- Dog that doesn’t hunt
- Discreet separation of gen and hand rolled
- The edge is much closer to the norm
- You still have to maintain the generated code

ACIDity

- Robustness
- Pretty much handled by the underlying database and data access technologies

Application Stays Agnostic To...

- Connection style
- Access technology

TRANSACTIONING

- Commit or Rollback

This is ***not***....

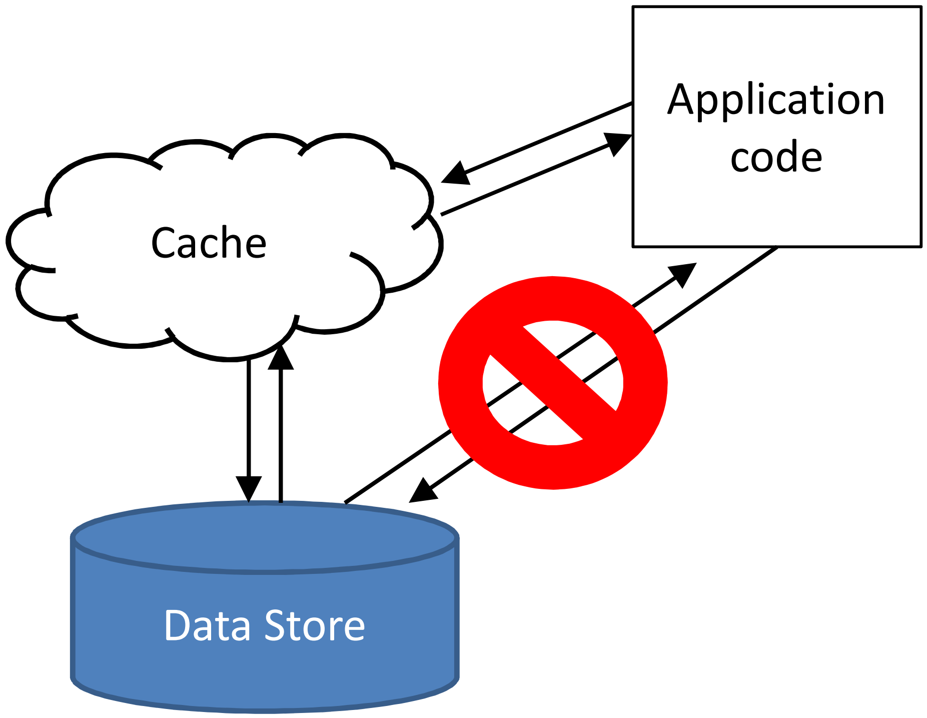
Unit Of Work

Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.

(Fowler, PEAA)

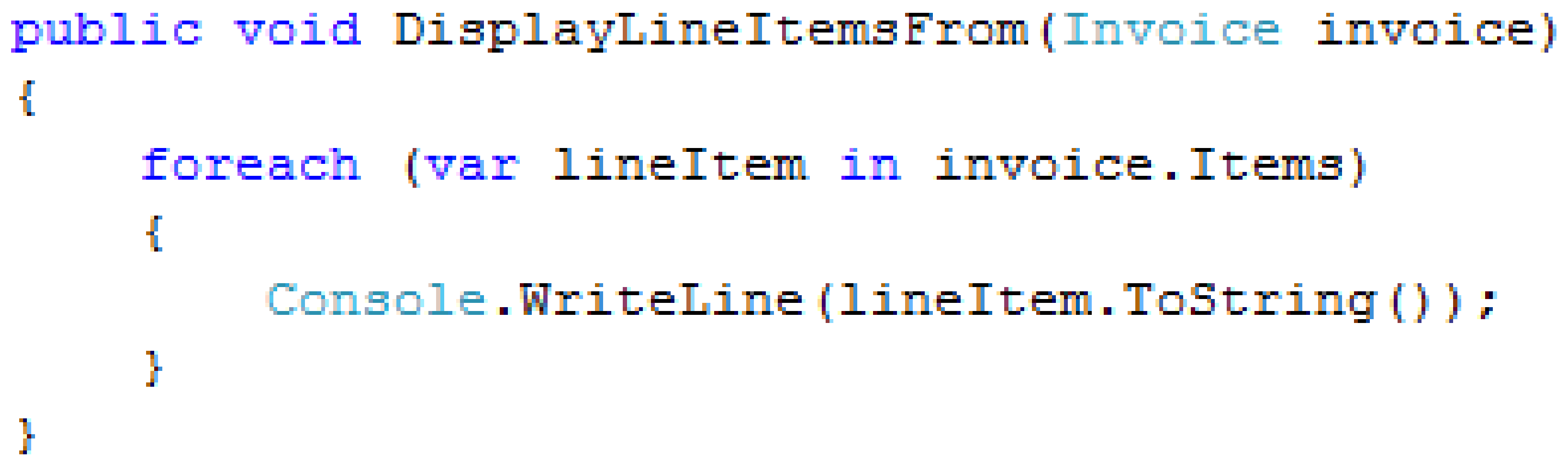
CACHING

- Should be hidden from the main application
- Should be configurable



LAZY LOADING

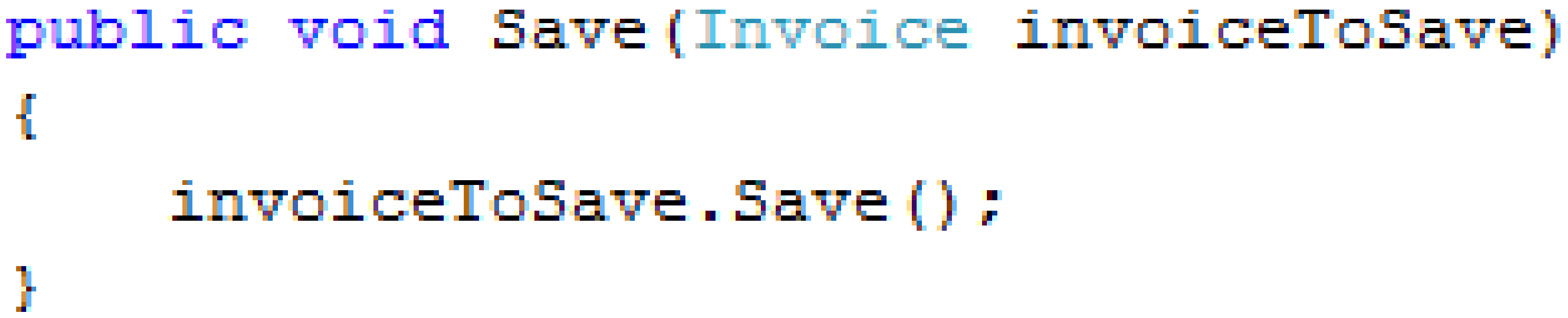
- Should be hidden from the main application
- Implicitly occurring
- Should be configurable



```
public void DisplayLineItemsFrom(Invoice invoice)
{
    foreach (var lineItem in invoice.Items)
    {
        Console.WriteLine(lineItem.ToString());
    }
}
```

PERSISTENCE IGNORANCE

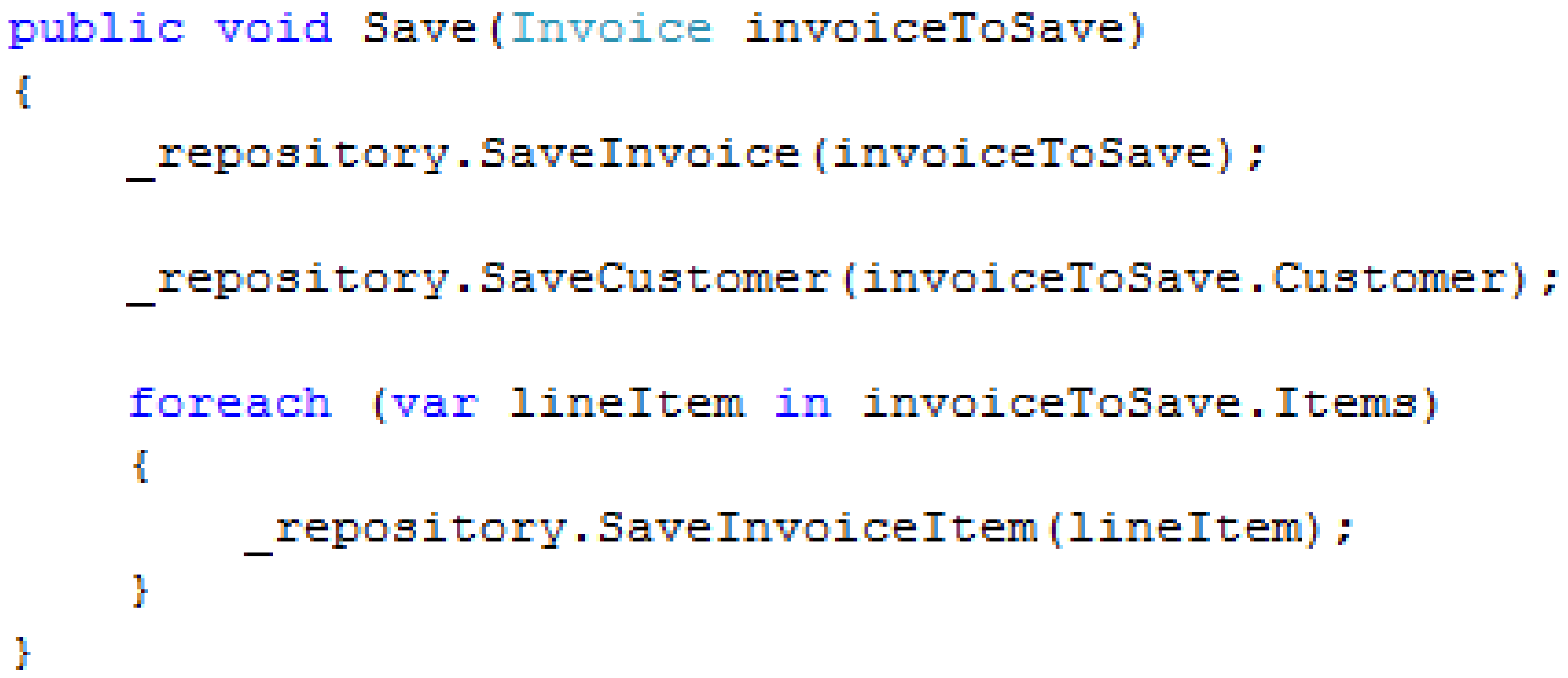
•Objects should have no knowledge about how they are persisted



```
public void Save(Invoice invoiceToSave)
{
    invoiceToSave.Save();
}
```

Persistence By Reachability

•Child objects should implicitly be traversed and persisted when saving the parent object



```
public void Save(Invoice invoiceToSave)
{
    _repository.SaveInvoice(invoiceToSave);

    _repository.SaveCustomer(invoiceToSave.Customer);

    foreach (var lineItem in invoiceToSave.Items)
    {
        _repository.SaveInvoiceItem(lineItem);
    }
}
```

Versioning & Deployment Story

- Versioning of the mapping between database and objects
- Versioning of any SQL that is required
- Ideally, the ability to include as separate items in source control
- Deployment of database DDL
- Deployment of data access components (SQL, SPs, mappings, etc.)

SUMMARY

A **good** ORM will save you in development ***and*** maintenance effort.

It will also provide you with capabilities that you ***will not have built*** otherwise.

Resources

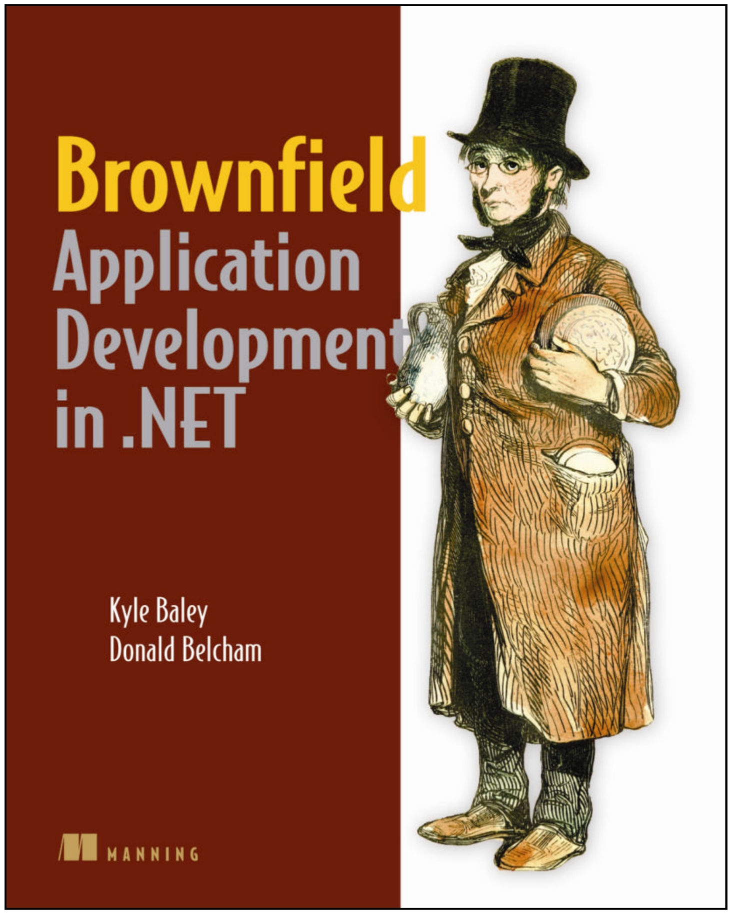
www.igloocoder.com

www.hibernate.org

www.nhforge.org

nhusers Google Group

donald.belcham@
igloocoder.com


Brownfield Application Development in .NET

Kyle Baley
Donald Belcham

MANNING

igloocoder.com
CONSULTING INC.