LINQ @ icoodb

Jb Evain
Mono Team

jbevain@novell.com

SHOCKING SLIDES

```
SELECT P.ADDRESS
WHERE P.AGE > 30
FROM PERSON P
ORDER BY P.NAME
```

```
engine.Select (
  “Partners[(count(Partners) > 0) or
(Name = 'p3')].Name”);
```

```
List<Address> addresses =
   OPathQuery<Address>()
      .Where("City = ? AND State = ?")
      .Sort("ZipCode ASC, StreetLine1
             ASC")
    .Parameters (someCity, someState)
    .GetList (engine);
```

```
session.createQuery(
  “from Cat as cat
   inner join cat.mate as mate
   left outer join cat.kittens as
    kitten”).list();
```

```
IQuery query = container.Query ();
query.Constrain (typeof (Person));
query.Descend (“Age”)
      .Constrain (30)
      .Greater ();
query.Descend (“Name”)
      .OrderAscending ();
```

?

LINQ

Language INtegrated Queries

```
static IEnumerable<Address>
GetAncientPeopleAddresses ()
{
  return from Person p in store
         where p.Age > 30
         orderby p.Name
         select p.Address;
}
```

```
return from Person p in store
       where p.Age > 30
       orderby p.Name
       select p.Address;
```

```
SELECT P.ADDRESS
WHERE P.AGE > 30
FROM PERSON P
ORDER BY P.NAME
```


SQL !!!
YOU MAKE
KITTY SCARED

```
return from Person p in store
       where p.Age > 30
       orderby p.Name
       select p.Address;
```

```
return store
        .Where (p => p.Age > 30)
        .OrderBy (p => p.Name )
        .Select (p => p.Address);
```

```ruby
return store
        .find_all { |p| p.Age > 30 }
        .sort_by { |p| p.Name }
        .map { |p| p.Address };
```

(-- F# --)

```
let results = store
|> List.filter (fun p -> p.Age > 30)
|> List.sort (fun p ->
       string.Compare p.Name)
|> List.map (fun p -> p.Address);;
```

```
return store
        .Where (p => p.Age > 30)
        .OrderBy (p => p.Name )
        .Select (p => p.Address);
```

```
store.Where (p => p.Age > 30)
```

```
Enumerable.Where (
   store,
   (Person p) => {
      return p.Age > 30;
   });
```

```
static class Enumerable {

  public static IEnumerable<T>
    Where<T> (
      this IEnumerable<T> coll,
      Func<T,bool> selector)
  {
    foreach (T item in coll)
      if (selector (item))
        yield return item;
  }
}
```

That's LINQ to Objects

LINQ to *

```
store.Where (p => p.Age > 30)
```

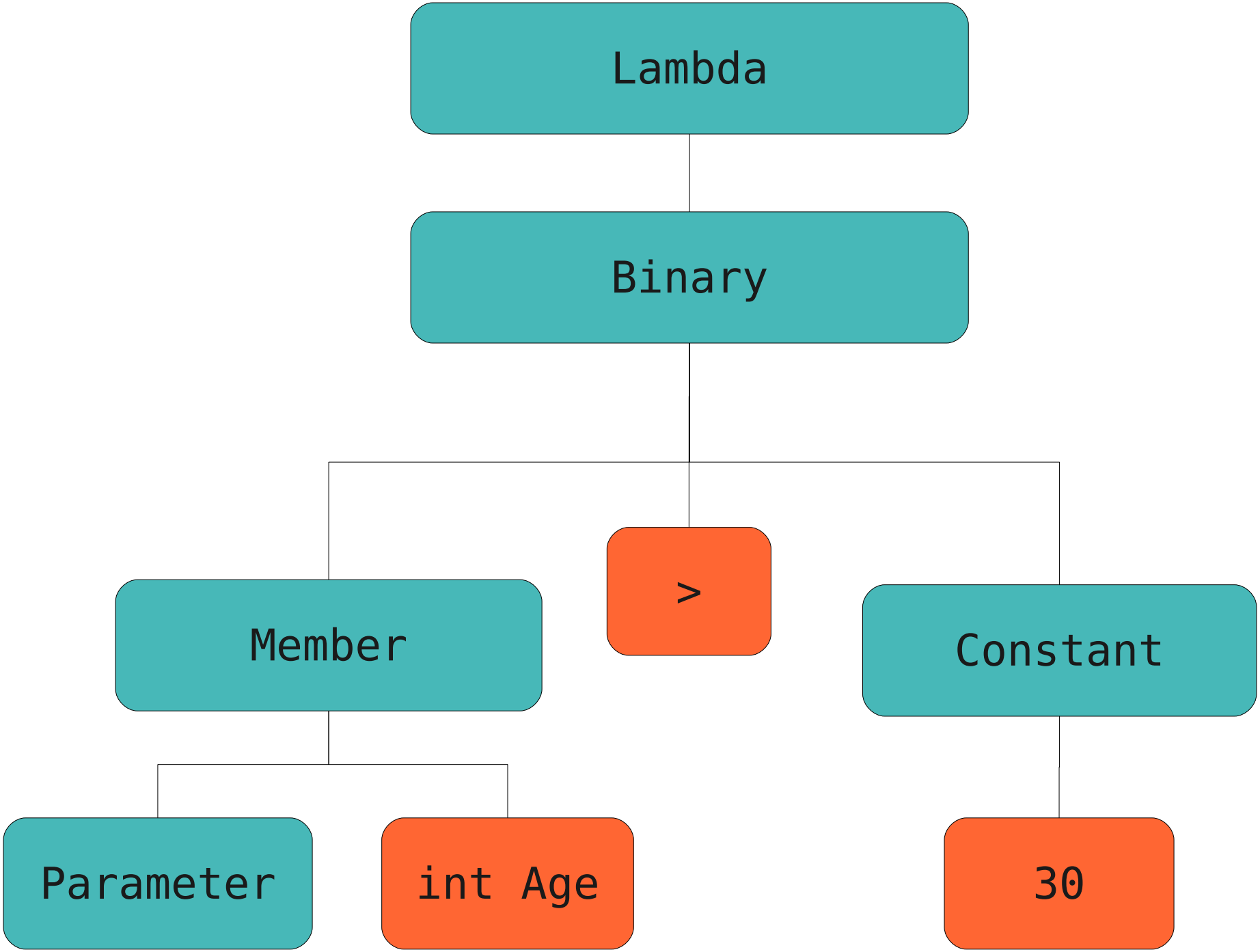
```
Enumerable.Where (
   store,
   (Person p) => {
      return p.Age > 30;
   });
```

```
static class Enumerable {

  public static IEnumerable<T>
    Where<T> (
      this IEnumerable<T> coll,
      Func<T,bool> selector)
  {
    foreach (T item in coll)
      if (selector (item))
        yield return item;
  }
}
```

```
static class LinqProvider {

  public static ICustom<T>
    Where<T> (
      this ICustom<T> coll,
      Expression<Func<T,bool>> tree)
  {
    // do something with the tree ..
  }
}
```

```
using System.Linq.Expressions;
using Exp = Expression;

var p = Exp.Parameter (
          typeof (Person));

LinqProvider.Where (
 store,
 Exp.Lambda<Func<Person,bool>> (
    Exp.GreaterThan (
      Exp.Member (
        p,
        typeof (Person)
          .GetProperty (“Age”)),
      Exp.Constant (30))));
```

```
store.Where (p => p.Age > 30)
```


Lambda
Binary
Member
>
Constant
Parameter
int Age
30

LINQ to db4o

```
using Db4objects.Db4o;
using Db4objects.Db4o.Linq;
// ..
IObjectContainer store;
// ..
IEnumerable<Person>
GetAncientPeople ()
{
  return from Person p in store
         where p.Age > 30
         orderby p.Name
         select p;
}
```

```
IQuery query = container.Query ();
query.Constrain (typeof (Person));
query.Descend (“Age”)
      .Constrain (30)
      .Greater ();
query.Descend (“Name”)
      .OrderAscending ();
```

```
return from Person p in store
       where p.Age > 30
        groupby p.Employer into g
        select new {
            Employer = g.Key,
            Employee = g };
```

(supposedly) no cats have been hurt in the making of this presentation.

questions ?