Scoping ActiveRecord

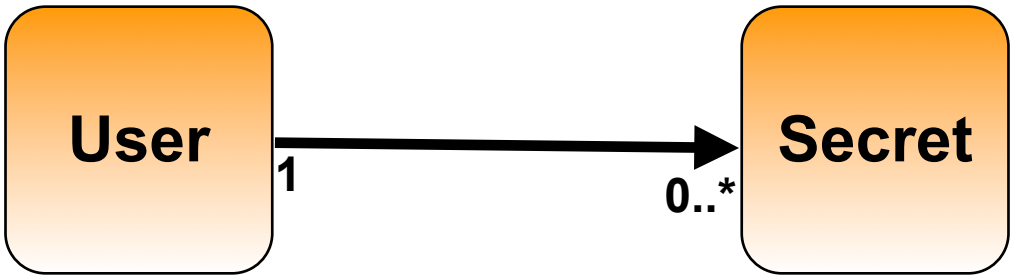
Marcus Ahnve

Valtech AB

About Me

- **Started out with Smalltalk**
  - Big Seaside fan as well
- **Long time J2EE developer**
- **Have used Ruby on and off since 2000**
  - Thanks to the Pragmatic Programmers and Anders Bengtsson
- **Have only used Rails in private projects**
- **Evangelizing within Valtech**

Demo Application



The problem

```
def index
    @secrets = Secret.find(:all, :conditions => ["user_id=?", 1])
end

def new
  Secret.create(:name => rand(10000), :user_id => 1)
  redirect_to :action => "index"
end
```

Enter with_scope

- **Scope parameters to method calls within the block.**
- **Takes a hash of method_name => parameters hash.**
- **method_name may be :find or :create.**
- **:find parameters may include**
  - :conditions
  - :joins
  - :include
  - :offset
  - :limit
  - :readonly
- **:create parameters are an attributes hash**

With scoping

```
def index
  Secret.with_scope(:find => {:conditions => ["user_id=?", 1]},
                    :create => {:user_id => 1} ) do
    @secrets = Secret.find(:all)
  end
end

def new
  Secret.with_scope(:find => {:conditions => ["user_id=?", 1]},
                    :create => {:user_id => 1} ) do
    secret = Secret.create(:name => rand(10000))
    if(secret.new_record?):
      flash[:notice] = "Could not create Secret"
    end
  end
  redirect_to :action => "index"
end
```

DRY ...

```
@@validUser = {:find => {:conditions => ["user_id=?", 1]}, :create => {:user_id => 1}}

def index
  Secret.with_scope(@@validUser) do
    @secrets = Secret.find(:all)
  end
end

def new
  Secret.with_scope(@@validUser) do
    secret = Secret.create(:name => rand(10000))
    if(secret.new_record?):
      flash[:notice] = "Could not create Secret"
    end
  end
  redirect_to :action => "index"
end
```

What about filters?

valtech

the currently logged in user, any calls involving users inside this block are automatically scoped for you. Wrap actions in a before_filter() and you end up with not only a cleaner code base but much less of a chance of *forgetting* to scope your database queries by account (which would result in users seeing other users outside of their accounts—not good!).

From Rails Recipes

before_filter?

- **Called before Controller invocation**
- **Method**
- **Block**
  - Current controller is passed as a parameter
- **Class**
  - filter() method called
- **But there is no way to use this for our purposes**

What about around_filter?

- **around_filter is not your regular AOP around filter**
- **Implemented with before() and after()**
  - No way to get the block syntactically right

A Better around_filter

```
class ScopingFilter
  def around
    DomainObject.with_scope(…) do
      yield # passes control to controller
    end
  end
end
```

Thanks to Jon Tirsen

So what do we have today?

- **scoped_access plugin**
- **Uses around_filter in ways God did not intended**
  - Messes around with internals
- **But it works**

The DRY:d up code

valtech

```
around_filter ScopedAccess::Filter.new(Secret,:mine)

def mine
   {:find => {:conditions => ["user_id=?", 1]}, :create => {:user_id => 1}}
end

def index
    @secrets = Secret.find(:all)
end

def new
  secret = Secret.create(:name => rand(10000))
  if(secret.new_record?):
    flash[:notice] = "Could not create Secret"
  end
  redirect_to :action => "index"
end
```

Restart Lighttpd after installing a plugin