Hiring Project

Intallation

1. Preparation

Clone Project

Use GitHub Website - Fork and Clone Project from: https://github.com/YourSole/hiring-project
Clone the project to GitHub folder called: hiring-project

2. Vagrant

Download and install vagrant from: https://www.vagrantup.com/downloads.html
Locate the root of hiring-project folder: ~/dai/GitHub/hiring-project
Run Vagrant installation process with terminal

```
vargrant up
```

It may take about 15 minutes to finish

The result should look like this

```
    default: To start the app on on http://localhost:7653
    default: vagrant ssh
    default: cd /vagrant/Edge-HiringProject
    default: plackup -p 7653 -R ./lib bin/app.psgi
==> default: Running provisioner: shell...
    default: Running: inline script
    default: To start the app on on http://localhost:7653
    default: vagrant ssh
    default: cd /vagrant/Edge-HiringProject
    default: plackup -p 7653 -R ./lib bin/app.psgi
```

Follow the instruction and ssh into the ubuntu instance

```
# Vagrant SSH
vagrant ssh
# Go to the root folder
cd /vagrant/Edge-HiringProject
# Run the project
plackup -p 7653 -R ./lib bin/app.psgi
```

The project should be online and accessable via port 7653. Confirmation below:

```
vagrant@vagrant-ubuntu-trusty-64:~$ cd /vagrant/Edge-HiringProject
vagrant@vagrant-ubuntu-trusty-64:/vagrant/Edge-HiringProject$ plackup -p 7653 -R ./lib
bin/app.psgi
Watching ./lib bin/lib bin/app.psgi for file updates.
HTTP::Server::PSGI: Accepting connections at http://0:7653/
```

2. Requirement

1. Create the new Edge::Order class (lib/Edge/Order.pm) to represent an order form submission.

The order.pm file would be just a normal class structure with all the variable needed. The order class itself doesn't need a constructor at the moment. The file should be something like this below:

```perl
package Edge::Order;

use Moose;

## Product variable
has 'product' => (
  is => 'ro',
  isa => 'Str',
  required => 1,
);

## Price Variable
has 'price' => (
  is => 'ro',
  isa => 'Int',
  required => 1,
);

# Quantity variable
has 'quantity' => (
  is => 'ro',
  isa => 'Int',
  required => 1,
);

# Total variable
has 'total' => (
  is => 'ro',
  isa => 'Int',
  default => sub {
    my $self = shift;
    return $self->quantity * $self->price;
  },
);

no Moose;
__PACKAGE__->meta->make_immutable;
```

2. Implement the "orders" attribute in the Edge::Customer class (lib/Edge/Customer.pm).

Psudocode for Java would be:

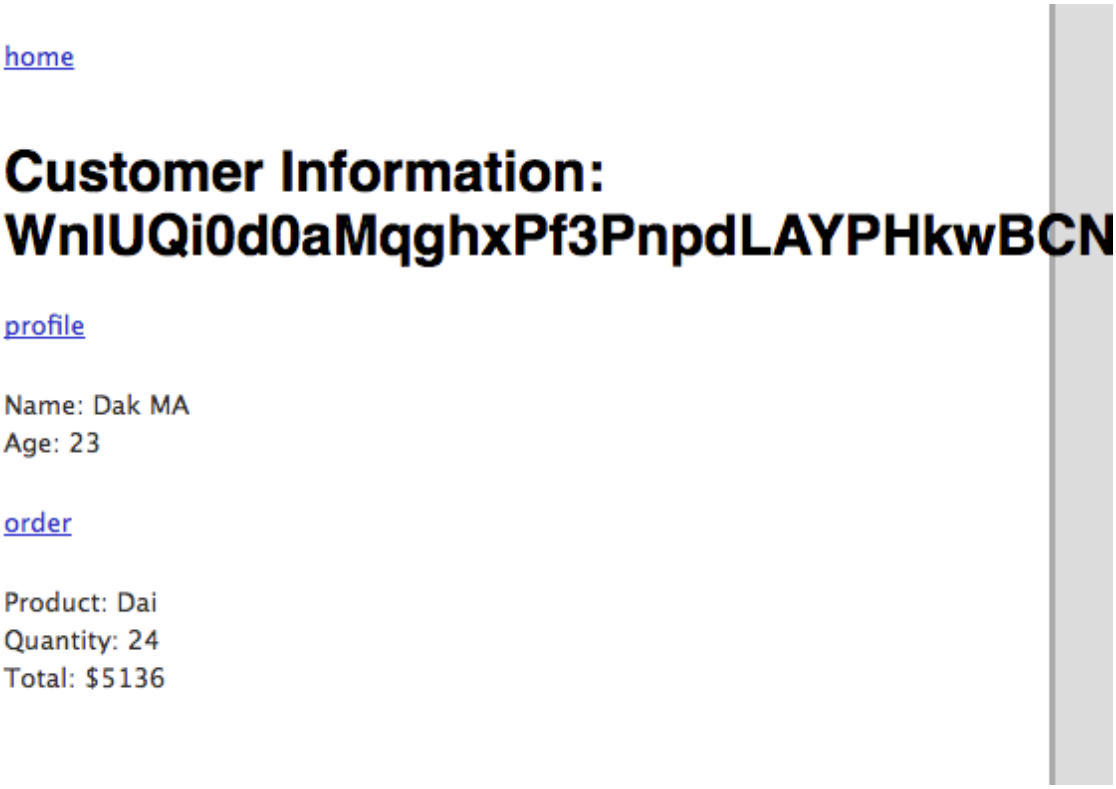
1. Accessing the database
2. Run query
3. Get resultlist and start ilterate through it and covert to an array of objects.
4. Return an array of the result.

I found it works the same way in Perl/Dancer2/DBIC

```
has 'orders' => (
  is => 'ro',
  isa => 'ArrayRef[Edge::Order]',
  lazy => 1,
  default => sub {
    my $self = shift;
    my @orders;
    # Preparing the resultset from database
    my $order_rs = $self->schema->resultset('FormSubmission')->search(
        {
          'data' => \["->>'form' = 'order'"],
        },{
          # Newest order on top
          'order_by' => { -desc => 'id' },
        },
    );

    # Run the query and start getting data
    while (my $order = $order_rs->next) {
      # Convert from HASHRef to data
      my $awsome = $order->data;
      # Push the data into a new Order object
      my $order_fetch = Edge::Order->new(price => $awsome->{price}, product => $awsome->
{product}, quantity => $awsome->{quantity});
      # Push the data into an array
      push @orders, $order_fetch;
    }
    # Return an array reference
    return \@orders;

  },
);
```

3. The customer view (views/customer.tt) should now display the profile and order form data from Edge::Customer for the current session.

The program now should look like this:



4. BONUS: pre-fill the profile form with the current session values from Edge::Customer

My original thought was to use Session::Cookie which I believe that is what you want me to do.

First solution:

1. Create a session in HiringProject.pm
2. Load data of the customer object on to the session object
3. Use the session data to fill profile page

```
# Turn session on
set session => 'YAML';

get '/form/:form_name' => sub {
...
template 'form' => {
            'form' => $form_name,
            'title' => $form->data->{title},
            'form_description' => $form->data->{description},
            'form_fields' => $form->data->{fields},
            # Use the session to call out customerinfo
            'customerinfo' => sub {
              if ($form_name eq 'profile') { return session('customer')->profile_form }
            }

get '/customer' => sub {
    my $customer = Edge::Customer->new( schema => schema('edge'), id => session->id );
    # Give the session data from fetching customer
    session customer => $customer;
```

Second solution:

1. Check if it is the profile form
2. Get the data from the database
3. Check if the customer data is availabe
4. Load it into the template

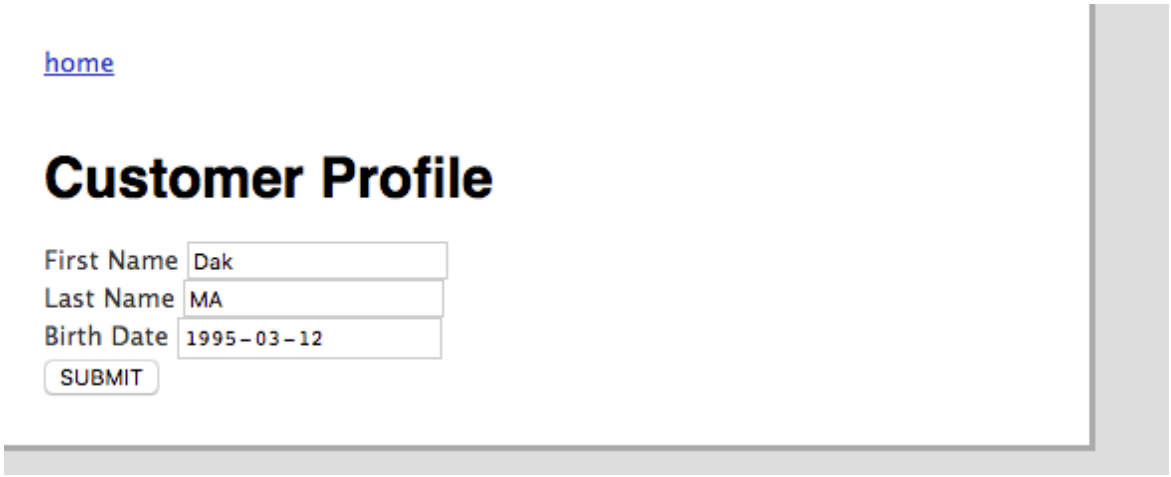
```
get '/form/:form_name' => sub {

    my $schema = schema 'edge';
    my $form_name = route_parameters->get('form_name') || '';
    my $form = $schema->resultset('Form')->search({
                'data' => \"->>'name' = '$form_name'"
               })->single;

    if ($form) {



        # Get user data
        my $profile_form = $schema->resultset('FormSubmission')->search(
                        {
                          'data' => \["->>'id' = ?", session->id],
                          'data' => \["->>'form' = 'profile'"],
                        },
                        {
                          'order_by' => { -desc => 'id' },
                        },
                      )->first;
        # Check if user data is filled
        # Check if the form is profile
        if ($profile_form && $form_name eq 'profile') {
            # Get the data into an array
            my @custom = ($profile_form->data->{fname}, $profile_form->data->{lname},
$profile_form->data->{birthdate});
            # Push the customer data to the template
            template 'form' => {
              'form' => $form_name,
              'title' => $form->data->{title},
              'form_description' => $form->data->{description},
              'form_fields' => $form->data->{fields},
             # Added a function to loop though the customer data
              'customerinfo' => sub { my $first = shift @custom; return $first},
            };
        } else {
        template 'form' => {
          'form' => $form_name,
          'title' => $form->data->{title},
          'form_description' => $form->data->{description},
          'form_fields' => $form->data->{fields},
        };
      }
```

```
    } else {
        redirect '/';
    }
};
```

File form

```
# line 12 added value="[% customerinfo %]"
<label>[% label %]</label> <input type="[% type %]" name="[% name %]" value="[% customerinfo %]"><br>
```

The result looks like the below:



5. BONUS: implement the 'age' attribute in Edge::Customer (hint: use Time::Piece)

To implement age, I use Time::Piece on the date object, get the year from their birthdate and subtract it by localtime->year

```
has 'age' => (
  is => 'ro',
  isa => 'Int',
  lazy => 1,
  default => sub {
    my $self = shift;
    my $birthdate = $self->profile_form->{birthdate};
    ## Get year out of birthdate
    my $year = Time::Piece->strptime($birthdate, "%Y-%m-%d")->year;
    ## Condition to show nothing till user input their birthdate
    if ($self->name eq '--not set--') {
      return 0;
    } else {
      my $age = localtime->year - $year;
      return $age;
    }

  },
);
```

The result should look like this:

home

Customer Information: WnlaLgLNF284C2CKboUVvQD8DePrFptd

profile

Name: Dai Nguyen
Age: 23

order

Product: Product 3
Quantity: 4
Total: $200

Product: Product 2
Quantity: 4
Total: $96

Product: Product 1
Quantity: 4
Total: $8