Evaluation of a Haskell Web Framework

Junaid Rasheed

The Project

- Evaluate a Haskell Web Framework
  - Viable?
  - Ready for Production?
  - Worth it?

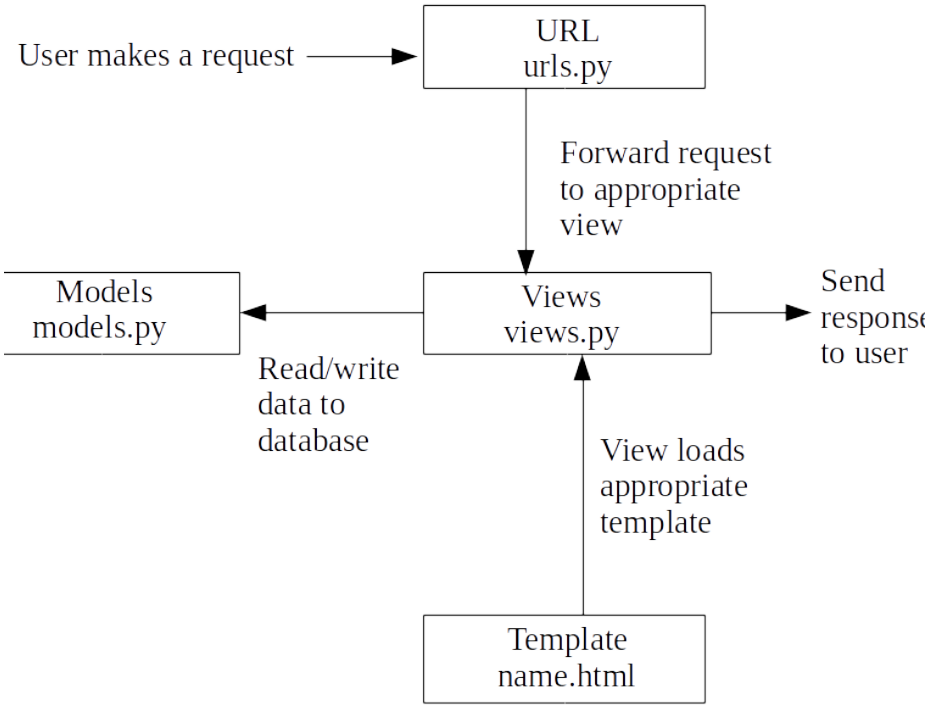
Our Hypothesis

- The Haskell framework should be faster
  - GHC compiles to code to native machine instructions
  - Lazy evaluation speeds up code execution
- But..
  - Does the framework produce a reliable, fast website?
  - Is developing in the framework viable? Refactoring? Testing?

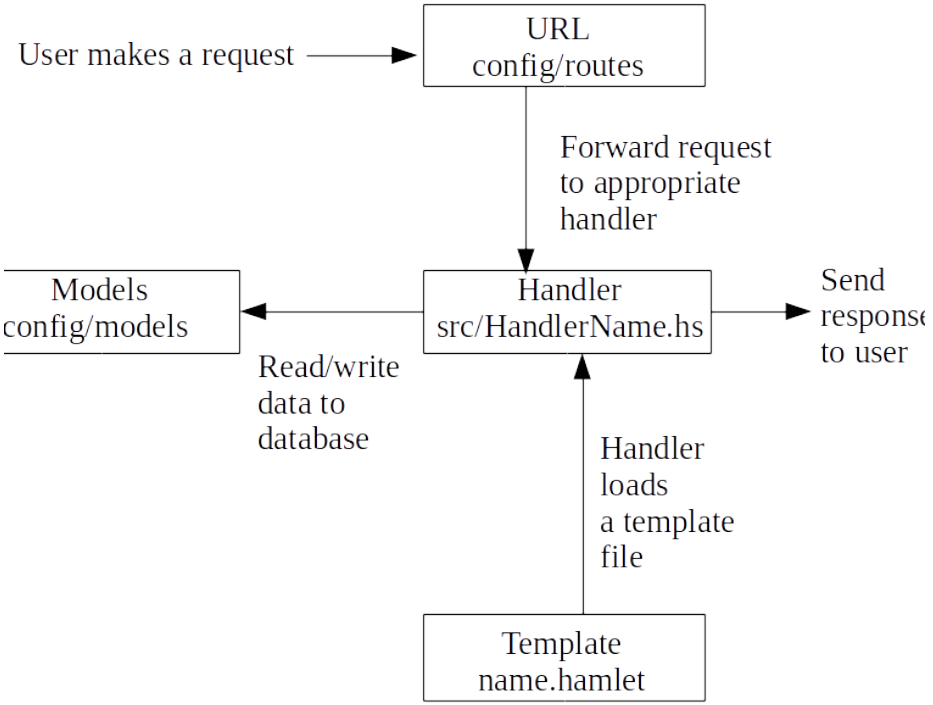
The Frameworks

- Yesod and Django
- Both are modular, ‘batteries included’ frameworks
- Django is very popular
- Some sites do use Yesod

Overview: Django



Overview: Yesod



Learning the Frameworks

- Haskell Programming From First Principles
- Developing Web Applications with Haskell and Yesod
- Django Documentation
- Haskell Library Documentation
- Forums, chat rooms

Wire

- A twitter clone was created, named Wire.
- Users can
  - Create accounts
  - Post messages
    - Including #hashtagged messages
  - Follow each other
  - Search for messages/users

Continuous Integration

- GitHub Repositories integrated with Travis CI
- Tests ran after every commit pushed to GitHub
- E-mail received if a test fails

Running the Experiments

- Sites were hosted on identical Amazon EC2 instances
- We ran page load speed tests, load tests, and analysed resource usage

Page Load Speeds

Page	Average Speed in Yesod (ms)	Average Speed in Django (ms)
Home Page	511.00	753.33
Search Page	517.33	756.33
Login Page	443.67	821.33
Signup Page	490.33	764.00
Creating an Account	504.33	748.67
Logging in to an Account	547.33	722.67
Logging out	510.33	761.33
Current user's Profile Page	617.00	930.00
Other user's Profile Page	651.33	908.67
Creating a Message	679.33	848.67
Search for a Message	513.33	766.33
Search for a User	519.00	756.67

- Tests repeated 3 times
- Yesod is ~200ms faster

Load Tests

Page	Yesod (s)	Django (s)
Home	4.96	5.54
Profile 1st Run	5.05	4.94
Profile 2nd Run	4.97	5.13
Average	4.99	5.20

- Used RedLine13 for these tests
- 80 users loaded a given URL
- Test normally took 25 seconds to complete
- Yesod transferred less data, ~6.46MB per test vs Django's 18.19MB per test

Resource Usage

- Similar resource usage on both servers
- Yesod server used 109MB of RAM, Django server used 125MB of RAM

Debugging

- Yesod’s type safety does prevent some errors
- No implicit type inference in Haskell
- Haskell debug messages are also very useful
- Tested how useful this is simulating errors

Saving Form Data

```
(Entity userId _) <- requireAuth -- get the user id
((result, _), _) <- runFormPost $ messageForm userId -- get the form data
case result of
  FormSuccess message -> do -- if it's a valid form, get the message
    -- _ <- runDB . insert $ message -- original line, insert message
    _ <- runDB . insert $ result -- new line, insert form data
```

```
form = NewWireForm(request.POST) -- get the form
if form.is_valid():
  if request.user.is_authenticated:
    message = form.cleaned_data['message'] -- form.cleaned_data is a map of form values
    try:
      # Message.objects.create(message_text=message, ..) # original line, store the message
      Message.objects.create(message_text=form.cleaned_data, ..) # changed line 1, store
    ↪ form values
      # Message.objects.create(message_text=form, ..) # changed line 2, after previous line
    ↪  passed, store form object
```

Saving Form Data

- Haskell compiler caught the error
- Python converted the form data into a string, and saved it into the database
- Tests should catch this

Test Limitations

- Free Amazon AWS tier limited the tests we could do
- Only a weak server was available
- Powerful servers would have affected previous tests

Conclusion

- Yesod is a production ready framework
- Faster than Django in our tests
- Yesod is recommended for:
  - Experienced Haskell Developers
  - Developers willing to learn Yesod
  - Developers concerned with reliability