Internet Personal Assistant

The Brewing of an IPA

Stephen Grabowski
Michael Groeneman
Anya Johnson
Max Lerner
Adrian Trunzo
Have You Ever Wanted...

• To not worry about setting your alarm?
• To know when to go get your friends at the airport?
• To automate your home?

...to have the boring things done for you?
What does the IPA do?

• Allows users to create and manage triggers.

• **Trigger**: a condition-action pair

• **Condition**: a truth-evaluable statement

• **Action**: do something!
Max's Trigger

If I'm late to algorithms and the temperature in my ZIP code drops below 40 degrees

Condition

Max's Trigger

Action

Text Me:
“It's 37 degrees out, grab your gloves... go now!”
If the cost of Apple stock drops below $400, with a P/E ratio below 15.

Buy 3% of my savings worth of AAPL.

Michael's Trigger
My friend's flight gets within 200 miles of MSP

Text Me: “Start driving over to the airport.”
Personal Assistants Today...

- Online Calendars
- Siri
- ifttt (our competitor)

... and now IPA.
The World is Full of Events (!)

- American Airlines Flight 1344 departed Dallas for Minneapolis at 6:05 PM.
- Let’s make a trigger that will let us know when it’s getting close to Minneapolis.

Volunteers?
The Brewing of the IPA

- Real-time Applicability
- Reliability
- Generality
- Extensibility
- Accessibility
1. check active triggers
2. get 'unchecked' triggers
3. Triggers
4. "Check Yo'self"
5. Evaluate
6. Evaluate
7. Get Attribute or Update
8. Scrape
9. Data
10. Attribute Value
11. T/F
12. True or False
13. Execute

To Interfaces

News Agent

Trigger Manager

Data Manager

Source

Condition

Operation

The Internet

PostgreSQL
The journey continues...

- A look under the hood
- Developer Tools
- User Interface
- Demo
Under the Hood...

IPA Core
1. check active triggers
2. get 'unchecked' triggers
3. Triggers
4. "Check Yo'self"
5. Evaluate
6. Evaluate
12. True or False
11. T/F
7. Get Attribute or Update
10. Attribute Value
8. Scrape
9. Data

The Internet

PostgreSQL
News Agent
The Front Desk

- Dispatches jobs to the rest of the core
- Manages input/output for user interfaces
- Translates between JSON and Python
- Stateless
- Asynchronous

Why...
- Stateless? - Accessibility
- Asynchronous? - Availability
1. check active triggers
2. get 'unchecked' triggers
3. Triggers
4. "Check Yo'self"
5. Evaluate
6. Get Attribute or Update
7. True or False
8. Scrape
9. Data
10. Attribute Value
11. T/F
12. Execute
Trigger Manager

• Handles adding/deleting/modifying triggers
• Checks active triggers when asked by News Agent

Why...
• Outsource condition evaluating?
  - Simplicity
  - Extensibility
1. check active triggers
2. get 'unchecked' triggers
3. Triggers
4. "Check it'self"
5. Evaluate
6. Evaluate
7. Get Attribute or Update
8. Scrape
9. Data
10. Attribute Value
11. T/F
12. True or False
13. Execute
To Interfaces

News Agent
Trigger Manager
Data Manager

Condition
Operation
Source
The Internet
Trigger
Action

PostgreSQL
Trigger

- Has a condition and action
- Contains timing data
- Load condition and action from DB (lazily)
- Evaluate condition and execute action
Condition

- Represented in a tree format
- Nodes can be:
  - Variables (Stock|AAPL|Price)
  - Operations (Sum, Greater Than, Distance)
  - Wolfram|Alpha (“population of China”)
- Condition ‘trees’ evaluate to true or false
- Nasty to represent in a DB!

```
{"nodeType": "operation", "arguments": [{"tableName": "Weather", "identifier": "55057", "nodeType": "variable", "attribute": "temperature"}, {"valueType": "Number", "nodeType": "constant", "value": "32"}], "value": "Greater"}
```
Source and Action

Source

- Custom representation of data from the Internet
- Knows how to grab information from the Web
- Knows how to store and retrieve data from Data Manager

Action

- Knows how to do something
- Can use variable information from database
1. check active triggers
2. get 'unchecked' triggers
3. Triggers
4. "Check Yo'self"
5. Evaluate
6. Evaluate
7. Get Attribute or Update
8. Scrape
9. Data
10. Attribute Value
11. T/F
12. True or False
13. Execute

Source

Condition

Operation

Trigger

Data Manager

To Interfaces

PostgreSQL
Why...

- Conditions as trees?
- Generality
- Performance
- Source object separate?
- Reliability
- Extensibility
News Agent

1. check active triggers

Trigger Manager

2. get 'unchecked' triggers

3. Triggers

4. "Check Yo'self"

Data Manager

5. Evaluate

6. Evaluate

7. Get Attribute or Update

8. Scrape

9. Data

Condition

10. Attribute Value

11. T/F

12. True or False

Operation

13. Execute

Source

The Internet

PostgreSQL
Data Manager
The Warehouse

- “All roads lead to Data Manager”
- Wrapper around a PostgreSQL database
- Translates between SQL and Python

Why...
- Make database calls go through Data Manager?
  - Simplicity
  - Extensibility
1. check active triggers
2. get ‘unchecked’ triggers
3. Triggers
4. “Check Yo’self”
5. Evaluate
6. Evaluate
7. Get Attribute or Update
8. Scrape
9. Data
10. Attribute Value
11. T/F
12. True or False
13. Execute

The Internet

PostgreSQL
Extending IPA

• Goal: Extensibility
  • Why? Possibilities are limitless.

• Strategy: Curation
  • Why? Security and reliability.

• Developers submit code for proposed sources, conditions, actions. Curators approve and add to system.
“So easy, an intro student could do it.”
Source

Data From Anywhere

Prerequisite: CS111
Source

Attributes
- What information does this source provide?
  - Fields

Updating Behavior
- How do we obtain this information?
  - Scrape()
  - Parse(scrapeResult)

Metadata
- Details we need to keep things running.
  - Verbose Name
  - Shelf Life
  - Identifier
Benefits of Good Ancestry

- (Rich) Type Checking
- Caching
- No Database Experience Necessary
- JSON Serialization Included
- Work like Django Models

Weather

Source

Inherits From

Data Model

Rich Type Checking
Serialization
DB Translation

Inherits From

Caching
Updating
Example: Airports

- For use with the Flight source.
  - e.g. Distance to Airport
- What information do we want?
  - Airport Name, City, Location
- How do we get the information?
  - FlightAware API. We send an XML request, FA grabs data and returns it.
  - Michael’s credit card gets charged (a little).
- Necessary Data Transforms
  - Convert latitude and longitude to a single PositionField.
- Metadata
  - Data current for one year

Source

Airport(“MSP”)

Code: MSP
City: Minneapolis, MN
Name: Minneapolis/St Paul Intl
Time Zone: CST6CDT
Location Coordinates: (-93.221,44.881)
Last Updated: 2012-02-20 18:07:10
class Airport(Source):
    # Fields
    code = StringField("Airport ICAO Code", maxLen=3)
    location = PositionField("Location Coordinates")
    city = StringField("City")
    name = StringField("Name")
    time_zone = StringField("Time Zone")

    # Metadata
    __verboasename__ = "Airport"
    __shelfLife__ = timedelta(days=365)
    __identifier__ = "code"

    # Updating Behavior
    def scrape(self):
        api = Client(flightapi_url, username=flightapi_username, password=flightapi_key)
        result = api.service.AirportInfo(self.code)
        return result

    def parse(self, raw):
        d = dict()
        d['city'] = raw['location']
        d['name'] = raw['name']
        d['time_zone'] = raw['timezone']
        d['location'] = [Decimal(raw['longitude']), Decimal(raw['latitude'])]
        return d
Operation

Compare Anything

Prerequisite: CS111
Operation

Evaluation
- How do we perform the operation
  - Evaluate(*args)

Metadata
- Details we need to keep things running.
  - Verbose Name
  - # of Arguments
  - Input/Return Types
class Distance(Operation):
    ''' Get the distance between two lists containing lat/lon coordinates'''

    # Metadata
    __verboasename__ = "Distance"
    num_arguments = 2
    parameter_types = (list,tuple)
    return_type = (decimal.Decimal,)

    # Evaluation
    def evaluate(self,*args):
        earthRadius = 3959

        lon1 = math.radians(args[0][0])
        lat1 = math.radians(args[0][1])
        lon2 = math.radians(args[1][0])
        lat2 = math.radians(args[1][1])

        d = math.acos(math.sin(lat1)*math.sin(lat2) + math.cos(lat1)*math.cos(lat2) * math.cos(lon2-lon1)) * earthRadius
        return decimal.Decimal(d)
Do Anything

Prerequisite: CS111
Execution: How do we perform the action

Attributes: What do we need to know to perform the action

Metadata: Details we need to keep things running.
Example: SendSMS

- Send an SMS to a specified number.
- What information do we need?
  - Cell Number (and Carrier)
    - e.g. 2405069235@sms.att.net
  - Message
- How do we do it?
  - Send an email through a carrier SMS gateway.
class SendSMS(ActionNode):
    # Required Information
    phoneNumber = LongField("Phone number to text", maxVal = 9999999999,
                            minVal=1000000000,required=True,default=None)
    carrier = StringField("Cellphone carrier",
                          choices=['AT&T', 'Sprint', 'Verizon', 'T-Mobile'],
                          required=True,default=None)
    message = StringField("Message to send", maxLen=500, required=True)

    # Metadata
    __verbosename__="Send a Text Message"

    # Execution Information
    def execute(self):
        server = smtplib.SMTP("smtp.gmail.com", 587)
        server.starttls()
        server.login(gmailAccount, gmailPassword)
        self.carrier = self.carrier.strip()
        server.sendmail(gmailAccount,
                        str(self.phoneNumber)+
                        '@'+phoneCarriersDict[self.carrier],
                        self.message)
class SendSMS(ActionNode):
    # Required Information
    phoneNumber = LongField("Phone number to text", maxVal = 9999999999, \ 
                           minVal = 1000000000, required=True, default=None)
    carrier = StringField("Cellphone carrier", \
                           choiceSet=['AT&T', 'Sprint', 'Verizon', 'T-Mobile'], required=True, default=None)
    message = StringField("Message to send", maxLen=500, required=True)

    # Metadata
    __verboasename__="Send a Text Message"

    ...
class SendSMS(ActionNode):
    # Required Information
    phoneNumber = LongField("Phone number to text", maxVal = 9999999999,
        minVal=1000000000, required=True, default=None)
    carrier = StringField("Cellphone carrier",
        choices=['AT&T', 'Sprint', 'Verizon', 'T-Mobile'], required=True, default=None)
    message = StringField("Message to send", maxLen=500, required=True)

    # Metadata
    __verbosename__="Send a Text Message"
class SendSMS(ActionNode):
    # Required Information
    phoneNumber = LongField("Phone number to text", maxVal = 9999999999,
                            minVal=1000000000, required=True, default=None)
    carrier = StringField("Cellphone carrier",
                         choiceSet=['AT&T', 'Sprint', 'Verizon', 'T-Mobile'],
                         required=True, default=None)
    message = StringField("Message to send", maxLen=500, required=True)

    # Metadata
    __verbosename__="Send a Text Message"
Interface...

Adrian Trunzo,
Master Interfacer

People We Don’t Know
Interface

IPA Core

(And non-Apple products too!)
Interface

Each interface has a common set of design principles:

• Usability
• Freedom
• Simplicity
Interface Tasks

• Create, Edit and Delete Triggers
• Make Conditions
• Associate Actions
• Set timing
• Monitor status of current triggers
• Manage user profile
• Explore the current state of the system
• Discover new sources and actions
The First Interface

• A website, with four simple spaces:
  • Welcome/Login page
  • Dashboard
  • Workshop
  • Profile

• What technologies are being used?
  • HTML 5
  • CSS
  • JavaScript (jQuery)
  • Python web server (Flask)
Add Trigger

Create a Condition

If

Operation

Constant
Variable
Meanwhile on Jeff’s Farm...

“Hey buddy... you wanna buy a hoofed mammal?”
(Larson 1984)
No Gnus...

...Bad News
Good gnus everyone!
Complex Triggers + Automatic Checking + Interface Independence + 3rd Party Extensibility

= Infinite Possibilities Available

= IPA
Thanks...

- Mike Tie
- DLN, for interrogating us
- ifttt
- The Django project
- Anya’s Parents
Thanks...

Jeff Ondich

Master of the Gnus Agent
APPLAUSE