Chapter 1:
In which Sep introduces himself and his problems
- accept lottery entries from mit students

- when I get a lottery entry from an MIT student,
  - put the entrant’s name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from mit students

- when I get a lottery entry from an MIT student,
  - put the entrant's name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from mit students

- when I get a lottery entry from an MIT student,
  - put the entrant's name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from MIT students

- when I get a lottery entry from an MIT student,
  - put the entrant’s name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from MIT students

- when I get a lottery entry from an MIT student,
  - put the entrant’s name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from MIT students

- when I get a lottery entry from an MIT student,
  - put the entrant’s name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from MIT students

- when I get a lottery entry from an MIT student,
  - put the entrant’s name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
- accept lottery entries from MIT students

- when I get a lottery entry from an MIT student,
  - put the entrant's name in a hat
  - thank the entrant

- after 1000 lottery entries,
  - pick an entry out of the hat
  - e-mail the winner
  - empty the hat
package edu.mit.media.social.lottery;

import javax.servlet.*;
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.sql.*;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class LotteryServlet extends HttpServlet {
  protected static final String USER_SESSION_KEY = "user";
  protected static final String ERROR_ATTRIBUTE_KEY = "error";
  public static void main(String[] args) throws Exception {
    Class.forName("org.postgresql.Driver");
    Server server = new Server(Integer.valueOf(System.getenv("PORT")));
    ServletContextHandler context = new ServletContextHandler()
      .setContextPath("/");
    server.setHandler(context);
    server.start();
    session = sessionFactory.openSession();
    session.beginTransaction();

    Statement statement = session.createStatement();
    String sql = "SELECT * FROM users WHERE email = " + email + " and password = " + password;
    ResultSet results = statement.executeQuery(sql);
    User user = null;
    while (results.next()) {
      String username = results.getString("username");
      String password = results.getString("password");
      String email = results.getString("email");
      Boolean confirmed = results.getBoolean("confirmed");
      String firstName = results.getString("first_name");
      String lastName = results.getString("last_name");
      Long id = results.getLong("id");
      String md = password;
      try {
        User user = new User(id, username, password, email, firstName, lastName, md);
        user.save(session);  // Save user to session
        user.saveMailMessage(session, "user", sessionFactory.buildSessionFactory());
        this.sendRedirect("index.jsp");
      } catch (Exception e) {
        throw new RuntimeException(e);
      }
    }
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String path = request.getPathInfo();
    if (path == null) {
      String viewPath = getJspPath("index.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    }
    else if (path == "signin") {
      String viewPath = getJspPath("signin.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    } else if (path == "signup") {
      String viewPath = getJspPath("signup.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    } else if (path == "logout") {
      session.invalidate();
      response.sendRedirect("/signin");
    } else if (path == "view") {
      String viewPath = getJspPath("view.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    } else if (path == "addUser") {
      String viewPath = getJspPath("addUser.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    } else if (path == "deleteUser") {
      String viewPath = getJspPath("deleteUser.jsp");
      RequestDispatcher dispatcher = request.getRequestDispatcher(viewPath);
      dispatcher.forward(request, response);
    }
  }

  public void processEmailMessage(User user) {
    if (!user.isConfirmed()) {
      String host = "smtp.gmail.com";
      String username = user.getEmail();
      String password = user.getPassword();
      Properties properties = new Properties();
      properties.put("mail.smtp.auth", "true");
      properties.put("mail.smtp.starttls.enable", "true");
      Session session = Session.getInstance(properties);
      Transport transport = session.getTransport("smtp");
      transport.connect(host, port, username, password);
      Transport.send(message);
    }
  }
}

@Entity
@Table(name = "lotteryentries")
public class LotteryEntry {
  @Id
  private Long id = 1;
  private String lotteryEntry;  // The lottery entry
  private String winningNumbers;  // The winning numbers
  private String prize;  // The prize
  // Other fields and getters
}

public class LotteryApplication {
  public static void main(String[] args) throws Exception {
    Class.forName("org.postgresql.Driver");
    Server server = new Server(Integer.valueOf(System.getenv("PORT")));
    ServletContextHandler context = new ServletContextHandler()
      .setContextPath("/");
    server.setHandler(context);
    server.start();
    session = sessionFactory.openSession();
    session.beginTransaction();

    Statement statement = session.createStatement();
    String sql = "SELECT * FROM users WHERE email = " + email + " and password = " + password;
    ResultSet results = statement.executeQuery(sql);
    User user = null;
    while (results.next()) {
      String username = results.getString("username");
      String password = results.getString("password");
      String email = results.getString("email");
      Boolean confirmed = results.getBoolean("confirmed");
      String firstName = results.getString("first_name");
      String lastName = results.getString("last_name");
      Long id = results.getLong("id");
      String md = password;
      try {
        User user = new User(id, username, password, email, firstName, lastName, md);
        user.save(session);  // Save user to session
        user.saveMailMessage(session, "user", sessionFactory.buildSessionFactory());
        this.sendRedirect("index.jsp");
      } catch (Exception e) {
        throw new RuntimeException(e);
      }
    }
  }
}

public class LotteryApplication {
  public static void main(String[] args) throws Exception {
    Class.forName("org.postgresql.Driver");
    Server server = new Server(Integer.valueOf(System.getenv("PORT")));
    ServletContextHandler context = new ServletContextHandler()
      .setContextPath("/");
    server.setHandler(context);
    server.start();
    session = sessionFactory.openSession();

    Statement statement = session.createStatement();
    String sql = "SELECT * FROM users WHERE email = " + email + " and password = " + password;
    ResultSet results = statement.executeQuery(sql);
    User user = null;
    while (results.next()) {
      String username = results.getString("username");
      String password = results.getString("password");
      String email = results.getString("email");
      Boolean confirmed = results.getBoolean("confirmed");
      String firstName = results.getString("first_name");
      String lastName = results.getString("last_name");
      Long id = results.getLong("id");
      String md = password;
      try {
        User user = new User(id, username, password, email, firstName, lastName, md);
        user.save(session);  // Save user to session
        user.saveMailMessage(session, "user", sessionFactory.buildSessionFactory());
        this.sendRedirect("index.jsp");
      } catch (Exception e) {
        throw new RuntimeException(e);
      }
    }
  }
}

public class LotteryApplication {
  public static void main(String[] args) throws Exception {
    Class.forName("org.postgresql.Driver");
    Server server = new Server(Integer.valueOf(System.getenv("PORT")));
    ServletContextHandler context = new ServletContextHandler()
      .setContextPath("/");
    server.setHandler(context);
    server.start();
    session = sessionFactory.openSession();

    Statement statement = session.createStatement();
    String sql = "SELECT * FROM users WHERE email = " + email + " and password = " + password;
    ResultSet results = statement.executeQuery(sql);
    User user = null;
    while (results.next()) {
      String username = results.getString("username");
      String password = results.getString("password");
      String email = results.getString("email");
      Boolean confirmed = results.getBoolean("confirmed");
      String firstName = results.getString("first_name");
      String lastName = results.getString("last_name");
      Long id = results.getLong("id");
      String md = password;
      try {
        User user = new User(id, username, password, email, firstName, lastName, md);
        user.save(session);  // Save user to session
        user.saveMailMessage(session, "user", sessionFactory.buildSessionFactory());
        this.sendRedirect("index.jsp");
      } catch (Exception e) {
        throw new RuntimeException(e);
      }
    }
  }
}
import community mit_students

listen to mit_students via http for lottery_entries

hat = []

on each lottery_entry do
    entrant = person from lottery_entry
    hat = hat append entrant
    notify entrant via http_response of "thanks! good luck."
end

on each 1000 lottery_entries do
    winning_entry = compute random_selection on lottery_entries
    notify person from winning_entry via email
    of "you won the lottery! congrats!"
    hat = []
end
The Dog Programming Language
Chapter 2: In which Sep describes Dog
Identifying people is hard

Talking to people is hard

Language integration is hard

Asynchronous state management is hard
Identifying people is hard
Data Layer

Users

Profiles

Relationships

User Model
create, read, update, delete, list, search

Session Controller

Account Controller

/signout.php
/signin.php
/create.php
/reset.php
good_students = PEOPLE FROM mit WHERE gpa > 3.5

responder = PERSON FROM response

sep = ME
Talking to people is hard
# = Listen to incoming emails - Ruby =
#
require 'net/imap'

imap = Net::IMAP.new(config["host"], config["port"], true)
imap.login(config["username"], config["password"])
imap.select("Inbox")
imap.uid_search(["NOT", "DELETED"]).each do |uid|
  # forward to your application logic
  imap.uid_copy(uid, "[Gmail]/All Mail")
imap.uid_store(uid, "+FLAGS", [:Deleted])
end
imap.expunge
imap.logout
imap.disconnect

# = Listen to incoming XMPP messages - Ruby =
#
require 'jabber4r/jabber4r'

begin
  session = Jabber::Session.bind(config["resource"], config["password"])
  myThread = Thread.current
  mlid = session.add_message_listener do |message|
    message.reply.set_body("Echo: #{message.body}").send
    myThread.wakeup if message.body=="shutdown"
  end
  Thread.stop
  session.delete_message_listener(mlid)
rescue Exception=>error
  puts error
ensure
  session.release if session
end
LISTEN TO students VIA email FOR assignments

NOTIFY program_committee VIA gtalk OF completed_review

ASK PEOPLE NEAR stata_center VIA sms TO do_a_dance
Language integration is hard
Worker Process
(DelayedJob, ZendQueue)

Job Database

Application
(Rails, PHP, NodeJS)

Scripts
(Python, Ruby, etc.)
<?php

class ClassificationJob {
    public function __construct($query, $location) {
        $this->query = $query;
        $this->location = $location;
    }

    public function perform() {
        $params = ""
        $params .= " -location " . $location
        $params .= " -query " . $query
        $classification = shell_exec("classify.py " . $params)
        // TODO: Continue processing
    }
}

$query = $_REQUEST['query']
$location = $_REQUEST['location']

// Started already with:
//    $worker = new DJWorker($options);
//    $worker->start();
DJJob::enqueue(new ClassificationJob($query, $location));
?>
IMPORT FUNCTION "classify.py" AS classify

COMPUTE classify ON data
Asynchronous state management is hard
post "/ticket" do
  verify_user_logged_in

  ticket = Ticket.new params["ticket"]
ticket.requester = session["current_user"]
ticket.save  # assigns unique id and saves to data layer
sendTicketEmail(Representatives.all, ticket)
end

post "/resolution" do
  verify_representative_logged_in

  ticket = Ticket.find(params["ticket_id"])
ticket.resolution = params["resolution"]
ticket.save

  requester = ticket.requester
sendResolutionEmail(requester, ticket)
end
# Help Desk Application

LISTEN TO users FOR tickets

ON ticket DO
  user = PERSON FROM ticket
  resolution = ASK representative VIA feed TO process ON ticket
  NOTIFY user OF resolution

END
Primitives for People

Human Functions

External Languages

Asynchronous Flows
Chapter 3:
In which Sep writes a Dog program
Newsfeed
#!/usr/bin/env dog

# ========================
# = Newsfeed Application =
# ========================

CONFIG default_community = "newsfeed"

newsfeed = community {
    relationship friends
}

posts = event {
    string body
}

LISTEN TO PEOPLE FROM mit VIA http FOR posts

ON post DO
    poster = PERSON FROM post
    friends = PEOPLE FROM poster’s friends
    NOTIFY friends VIA feed OF post
END
#!/usr/bin/env dog

# ======================  
# = Conference Reviews =  
# ======================  

IMPORT COMMUNITY conference
#!/usr/bin/env dog

# = Conference Reviews =
# ================

IMPORT COMMUNITY media_lab

reviewers = PEOPLE FROM media_lab WHERE status == "reviewer"
pc = PEOPLE FROM media_lab WHERE status == "program_committee"
IMPORT COMMUNITY media_lab

reviewers = PEOPLE FROM media_lab WHERE status == "reviewer"
pc = PEOPLE FROM media_lab WHERE status == "program_committee"

LISTEN TO PUBLIC VIA email AT "submissions@media.mit.edu" FOR submissions
#!/usr/bin/env dog

# = Conference Reviews =

import community media_lab

reviewers = people from media_lab where status == "reviewer"

pc = people from media_lab where status == "program_committee"

listen to public via email at "submissions@media.mit.edu" for submissions

on each submission do
#!/usr/bin/env dog

# ======================  
# = Conference Reviews =  
# ======================  

IMPORT COMMUNITY media_lab

reviewers = PEOPLE FROM media_lab WHERE status == "reviewer"

pc = PEOPLE FROM media_lab WHERE status == "program_committee"

LISTEN TO PUBLIC VIA email AT "submissions@media.mit.edu" FOR submissions

ON EACH submission DO
  reviews = ASK 3 reviewers TO review ON submission
#!/usr/bin/env dog

# ======================  
# = Conference Reviews =  
# ======================  

IMPORT COMMUNITY media_lab 

reviewers = PEOPLE FROM media_lab WHERE status == "reviewer"
pc = PEOPLE FROM media_lab WHERE status == "program_committee"

LISTEN TO PUBLIC VIA email AT "submissions@media.mit.edu" FOR submissions

ON EACH submission DO
    reviews = ASK 3 reviewers TO review ON submission
    NOTIFY pc OF reviews
IMPORT COMMUNITY media_lab

reviewers = PEOPLE FROM media_lab WHERE status == "reviewer"
pc = PEOPLE FROM media_lab WHERE status == "program_committee"

LISTEN TO PUBLIC VIA email AT "submissions@media.mit.edu" FOR submissions

ON EACH submission DO
  reviews = ASK 3 reviewers TO review ON submission
  NOTIFY pc OF reviews
END
Chapter 5:
In which Sep gets to the point
Decreased Cost, Increased Activity
New Paradigm for Programs
Programming Languages as an Art
www.dog-lang.org
Journal Transcription
#!/usr/bin/env dog

# =========================
# = Journal Transcription =
# =========================

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
    programming_languages CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO

    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf

END
#!/usr/bin/env dog

# = Journal Transcription =
# = ____________________________=

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
    programming_languages CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf

END
#!/usr/bin/env dog

# =========================
# = Journal Transcription =
# =========================

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
    programming_languages CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf

END
#!/usr/bin/env python

# = Journal Transcription =

import mechanical_turk, media_lab
import classify.py, "assemble.py"

turkers = PEOPLE FROM mechanical_turk
media_lab_students = PEOPLE FROM media_lab WHERE
    programming_langauges CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf

END
#!/usr/bin/env dog

# =========================
# = Journal Transcription =
# =========================

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
    programming_languages CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf
END
#!/usr/bin/env dog

# = Journal Transcription =

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
  programming_languages CONTAINS "LaTeX" AND
  status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
  pieces = COMPUTE segmentation ON files
    USING file_type = "pdf" AND split_on_white_space = true

  labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
    USING instructions = "Does this piece of text contain a math expression?"

  math_pieces = labeled_pieces["yes"]
  text_pieces = labeled_pieces["no"]

  transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
  transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
    USING mode = "math"

  searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
    USING path = "output"

  NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
    USING attachment = searchable_pdf
END
#!/usr/bin/env dog

# = Journal Transcription =

IMPORT COMMUNITY mechanical_turk, media_lab
IMPORT FUNCTION "classify.py", "assemble.py"

turkers = PEOPLE FROM mechanical_turk

media_lab_students = PEOPLE FROM media_lab WHERE
    programming_languages CONTAINS "LaTeX" AND
    status == "student"

LISTEN TO PUBLIC VIA email AT "journal@dormou.se" FOR journal

ON journal DO
    pieces = COMPUTE segmentation ON files
        USING file_type = "pdf" AND split_on_white_space = true

    labeled_pieces = ASK turkers TO classify ON pieces USING payment = 0.10
        USING instructions = "Does this piece of text contain a math expression?"

    math_pieces = labeled_pieces["yes"]
    text_pieces = labeled_pieces["no"]

    transcribed_text_pieces = ASK turkers TO transcribe ON text_pieces
    transcribed_math_pieces = ASK media_lab_students TO transcribe ON math_pieces
        USING mode = "math"

    searchable_pdf = COMPUTE assemble ON transcribed_text_pieces, transcribed_math_pieces
        USING path = "output"

    NOTIFY PERSON FROM journal VIA email OF "Your journal is transcribed!"
        USING attachment = searchable_pdf

END