# Implementing OpenID for Your Social Networking Web Site

By David Keener

http://www.keenertech.com

# Introduction

- Social networking sites are communities
- Communities consist of people
- Getting people to join your community is hard

- What if there was a technology that made it easy for people to join your community?

# Presentation Goals

By the time this presentation is over, you will….

- Understand how OpenID works from the user perspective
- Have a basic idea of how OpenID works behind the scenes
- Know how to implement OpenID for a web site using Ruby on Rails
- Have some perspective on how OpenID can be integrated into a social networking site

# Part 1: The Basics

# So, What Is OpenID?

- Single login, multiple web sites
- Simple and light-weight sign-on service
- Easy to use and deploy
  - Already supported in multiple languages
- An open standard
  - Changes based on community needs
- De-centralized identity verification
  - Nobody owns it
  - Nobody controls it
  - No single point of failure
- Free

# Hasn't This Been Done Before?

- Anybody remember "Windows Live ID" - alias ".NET Passport" and "Microsoft Passport Network?"
- There have been single-ID solutions from various vendors, but no universal acceptance … or adoption
- Probably only achievable by an open source standard that's not owned by any single vendor

# A Few Statistics…

- OpenID Identities: 120 million (07/07)
- AOL Identities: 63 million (05/07)
- Sites Supporting OpenID: 4500+ (07/07)

- Expecting: 250 million ID's and 15,000 supporting sites by end of 2007

# Why Do Users Need OpenID?

Um, it's the Holy Grail of the Internet…"one ID to rule them all."

- Users can login to many sites with a single ID
- No need to remember multiple user names & passwords
- Puts control of a user's ID in the hands of the user
- The user decides who manages their identity online
- Facilitates communication – think of Technorati linking to millions of blogs: Users don't want to create new accounts every time they respond

# Why Do Developers Want OpenID?

- Simplifies user management features for web sites
- Removes complexities associated with securely managing passwords
- Site specific hacks: “Login with your AOL screen name and get updates via AIM.”
- Accessibility for millions of potential users

# So, What Is an OpenID?

Well, it sounds cool. But what, exactly is an OpenID?

- It's a personal URL
- It references an "identity" and an "identity provider"
  - Ex. – openid.aol.com/davidkeener01
  - Ex. – dkeener.myopenid.com
  - Ex. – keenertech.com/dkeener (delegated)
- Users can choose the third party that will manage their online identity:
  - AOL
  - MyOpenID.com
  - Thousands of other sites

# What Can You Do With an OpenID?

Let's make it even simpler. An OpenID is a personal URL. This is what you can do with it:

- You can claim that you own it.
- You can prove that you own it.

Everything else evolves from this….

# So, How's It Work for the User?


First, you need a good Login page.

Make sure to give users some info on OpenID.

This sample Login page provides separate logins for OpenID and "AOL".

# Login Flow (User Perspective)


1. User provides OpenID to web site


2. Authenticate with Identity Provider


3. Re-direct user back to web site
   - Success: Go to desired destination
   - Failure: Back to Login page

** Yes, there's other complexities, but we'll talk about them later*

# Part 2: The OpenID Spec

# First, a Note About Modes

To support as many situations as possible, the OpenID spec includes two basic modes of operation….

- **Stateless** – So-called “dumb” mode; we don’t care about this mode – we’re not creating an OpenID-enabled toaster...

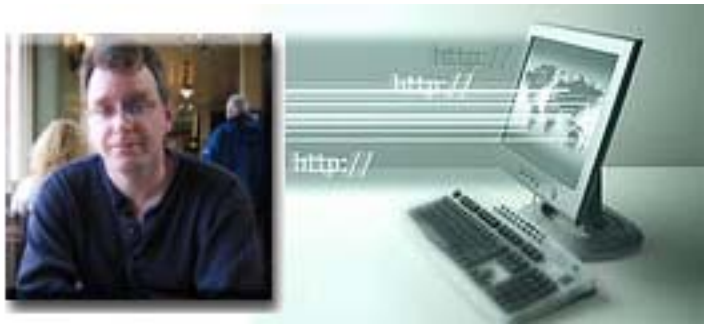
- **Stateful** – State is maintained between web server and OpenID Provider, allowing communications to be streamlined.

# OpenID Terminology

- **Identifier** – A URL owned by an End User.
- **End User** – The person who wants to prove their ownership of an Identifier to a Relying Party
- **Relying Party** – (formerly, “Consumer”) The web server that wants to verify an End User’s claim to own an Identifier
- **User-Agent** – The web browser of the End User
- **Identity Provider** – The OpenID Authentication Server contacted by a Relying Party to verify an End User’s ownership of an Identifier

# Our Players

For the discussion, here are our players in the OpenID process….




**User**
**(David Keener)**

**User-Agent**
**(the browser)**

WORLDS ENOUGH
SF and Fantasy ... Online

**Relying Party (Consumer)**
- Our sample social networking site

**Identifier**
- openid.aol.com/davidkeener01

AOL

**Identity Provider**
- openid.aol.com

# Behind the Scenes… (Part 1)

Here's what's really happening behind the scenes.

1. User provides their OpenID to a web site (the Relying Party).
2. Web Server verifies existence of Identity Provider (or delegate) by accessing identity-related HTML file.
3. Web Server and Identity Provider form an Association – cryptographic magic is done to create a shared secret so they can communicate securely.
    - One shared secret per Identity Provider
    - Stored locally to facilitate future communication
    - Expires periodically for security reasons

# Behind the Scenes… (Part 2)

4. Web Server re-directs User-Agent to Identity Provider for authentication, providing:
   - OpenID…….. openid.aol.com/davidkeener01
   - Trusted Root…. worldsenough.com
   - Return URL…... http://www.worldsenough.com/login/complete
     (Includes URL parameters to identify session, plus nonce)
5. The Identity Provider authenticates the claimed identity…
   - Login/Password, key fob, retinal scan, etc.
6. Identity Provider re-directs User-Agent to Return URL.
   - Result (Success, Failure, Cancel)
   - OpenID
   - Return URL
   - Cryptographic Magic (handle, signed fields list, signature)

# Behind the Scenes… (Part 3)

7. At Return URL, the Web Server takes action based on authentication result:
   - Failure: Back to Login page, with error message
   - Success: Go to next step…
8. (Optional) Get SREG information, if available
9. Re-direct user to appropriate destination

OpenID Sign-on Complete!

# OpenID Provider Details

The goal of OpenID is to make accessing web sites easier for users. Many providers support “ease-of-use” options:

- **Trusted Site Designation:** Provides automatic logins or access to SREG data if the user designates any site as a “trusted site.”
- **Auto-Logins:** If user has logged in during current browser session, subsequent OpenID logins will not be needed.
- **Remember Me:** Stores cookie allowing OpenID login to be remembered for future sessions.

# What's a Delegate?

Suppose you'd like to have your own identity, based on your own domain name (like keenertech.com/dkeener), but you'd rather not run your own OpenID Server….

- Ensure existence of a web site with your domain name
- Put an HTML file out on the web site
  - Proves you have rights to that URL
  - Ex: http://www.keenertech.com/dkeener/index.html
- Include some special HTML tags in the head section of the HTML page to:
  - Indicate who the real Identity Provider is
  - Indicate what identity is being delegated

# Delegate HTML Page

- OpenID: keenertech.com/dkeener
- URL: http://www.keenertech.com/dkeener/index.html

```
<html>
<head>
  <title>OpenID Verification: dkeener</title>
  <link rel="openid.server" href="http://www.myopenid.com/server" />
  <link rel="openid.delegate" href="http://dkeener.myopenid.com/" />

  <link rel="openid2.local_id“ href="http://dkeener.myopenid.com" />
  <link rel="openid2.provider“ href="http://www.myopenid.com/server" />

  <meta http-equiv="X-XRDS-Location“
       content="http://dkeener.myopenid.com/xrds" />
</head>
<body>
<p>OpenID Verification: dkeener</p>
</body>
</html>
```

# Part 3: Implementing OpenID

# Supporting OpenID in Rails

- You need to install the ruby-openid gem
  - Ex: gem install ruby-openid
- Now officially supports OpenID 2.0, as of December 5, 2007.
- For more information on ruby-openid:
  - http://www.openidenabled.com/ruby-openid

# Let's Implement OpenID in Rails

We're going to need the following files:

- Login Partials
  - apps/views/openid/_aol.rhtml
  - apps/views/openid/_openid.rhtml
- Login Page
  - apps/views/login/index.rhtml
- OpenID Controller
  - apps/controllers/openid_controller

# Login Partial

Just a basic HTML form, nothing exciting….

```
<fieldset>
<legend>Sign In Using OpenID</legend>

<%= start_form_tag :controller => 'openid', :action => 'login' %>
<input type="hidden" name="login_type" id="login_type" value="openid" />
<table>
<tr>
   <td>OpenID:</td>
   <td><input type="text" name="openid_url" class="openid" /></td>
   <td><%= image_submit_tag "button_login.jpg" %></td>
</tr>
</table>

</form>

</fieldset>
```

- Hidden field indicates whether the form is for an OpenID login or an “AOL” login.
- Posts to the “login” action of the OpenID Controller.

# Login Page

The Login page is equally exciting….

```
<h1>Login</h1>

<p><b>Worlds Enough</b> is pleased to offer you
   a variety of ways to login to our web site. You
   may use either OpenID or your AOL screen
   name to log in; in both cases, authentication is
   handled by your chosen identity provider.</p>

<%= render :partial => 'openid/openid' %>
<br />
<%= render :partial => 'openid/aol' %>
```

# The OpenID Controller (Part 1)

```
class OpenidController < ApplicationController
  layout nil
  require 'openid'

  def login
  end

  def complete
  end

  private

  # Get an OpenID::Consumer object. Will also create a store for
  # storing OpenID information in the application's "db" dir.

  def openid_consumer
    @openid_consumer ||= OpenID::Consumer.new(@session,
      OpenID::FilesystemStore.new("#{RAILS_ROOT}/db/openid"))
  end

end
```

- Layout is nil because this controller will not cause any views to be displayed
- Must have a “require” statement for OpenID.
- Note the private function “openid_consumer” which will be used by both the “login” and “complete” functions.

# The OpenID Controller (Part 2)

```
def login
  openid = params[:openid_url]
  login_type = params[:login_type]
  if login_type == "aol"
    openid = "openid.aol.com/" + openid
  end

  oid_res = openid_consumer.begin openid

  case oid_res.status
    when OpenID::SUCCESS
      return_url = url_for :action => 'complete'
      trust_root = url_for :controller => ''
      redirect_url = oid_res.redirect_url(trust_root, return_url)
      redirect_to redirect_url
      return

    when OpenID::FAILURE
      flash[:notice] = "Could not find OpenID server for #{openid}"
    else
      flash[:notice] = "An unknown error occurred."
  end
  redirect_to :controller=>"login", :action=>"index"
end
```

# The OpenID Controller (Part 3)

```
def complete
  oid_res = openid_consumer.complete params

  case oid_res.status
    when OpenID::SUCCESS
      session[:openid] = oid_res.identity_url
      session[:user_id] = User.check_user(response.identity_url)
      redirect_to :controller=>"admin", :action=>"index"
      return
    when OpenID::FAILURE
      if oid_res.identity_url
        flash[:notice] = "Verification of #{oid_res.identity_url} failed."
      else
        flash[:notice] = 'Verification failed.'
      end
    when OpenID::CANCEL
      flash[:notice] = 'Verification cancelled by the user.'
    when OpenID::SETUP_NEEDED
    else
      flash[:notice] = 'Unknown response from OpenID server.'
    end
    redirect_to :controller=>"login", :action=>"index"
  end
```

User.check_user looks up the user ID for the identity. If not found, then it creates a new user.

# Integration Recommendations

For your social networking site…

- **Accounts:** Associate OpenID logins with a user account.
- **Profile Page:** For first-time login, present a profile page (possibly populated with SREG data if available).
- **Security:** Add a layer of additional security for features involving money or access to critical private information.

# Part 4: Wrapping Up

# OpenID as a Building Block

OpenID solves the problem of “identity,” not “trust”…but think of the things that can be built on top of OpenID…

- **SREG:** Extension that allows Relying Parties to request simple registration info.
- **Trusted Extension:** Proposed extension to augment OpenID’s trust capabilities.
- **Whitelists:** A mechanism being discussed for identifying “responsible” Identity Providers.

# More Info About OpenID (Part 1)

- The official OpenID web site.
  - http://openid.net
- Resources for OpenID.
  - http://www.openidenabled.com
- Good article on implementing OpenID.
  - http://www.danwebb.net/2007/2/27/the-no-shit-guide-to-supporting-openid-in-your-applications
- A case study for OpenID-enabling an app.
  - http://www.plaxo.com/api/openid_recipe
- A good blog entry on OpenID adoption.
  - http://dev.aol.com/article/2007/openid_blog_part2

# More Info About OpenID (Part 2)

- Excellent 7-minute OpenID screencast.
  - http://leancode.com/openid-for-rails
- 8 OpenID resources for developers.
  - http://www.rubyinside.com/7-openid-resources-for-rails-developers-418.html
- An excellent book on OpenID, in PDF form:
  - The OpenID Book, by Rafeeq Rehman
  - www.openidbook.com/download/OpenIDBook-draft-15.pdf
- A negative article about OpenID. I really disagree with much of it, but it's certainly a good overview of OpenID criticisms.
  - http://www.idcorner.org/?p=161

# Summary

- OpenID removes a major entrance barrier for web site usage.
- Any social-oriented web site should be supporting OpenID.
- OpenID is great for users…it simplifies the login process and allows users to manage their own identities.
- Caution: With phishing possibilities, put a little extra security around monetary transactions and other critical actions.