[[1]](#footnote-2)

ASP.NET Membership and Personalization for OpenID

 (December 2008)

Idaho O. Edokpayi

# INTRODUCTION

O

penID is an open sourced, decentralized digital identity standard. OpenID allows users to use credentials entered at one site at any website that supports the technology. Users are allowed to maintain their digital identities at a central location and choose what personal information others are allowed to see. Websites are relieved of the burden of developing and maintain secure authentication schemes, are able to automatically query for user profile information, and are able to build more seamless user experiences. Security has the potential to be much better since users are required to enter personal information at fewer sites and websites can ask OpenID providers for more secure authentication protocols.

There are potential pitfalls however – Anyone is allowed to be an OpenID provider so relying parties – websites or other entities that use OpenID authentication to control access – must be careful. Also, malicious websites could potentially use phishing attacks to harvest users’ identifiers and credentials and would then have access to user information across a wide variety of sites. Finally, many larger companies have been criticized for being OpenID providers but not relying parties – critics think these companies are trying to stifle competition and extend their own brands to other websites. Security of identifiers provided by other OpenID providers is a legitimate concern but these companies often don’t allow identifiers provided by very large reputable competitors.

# Design

My class project (available at [www.whoisidaho.com/OpenIDDEMO](http://www.whoisidaho.com/OpenIDDEMO) ) is an ASP.NET web application that uses ASP.NET membership and personalization providers for authorization but uses OpenID for authentication. In my case, I used the DotNetOpenID library from the Google Code library to handle interactions with the OpenID protocol. Users enter their identifier (or click the button on the bottom of the page to directly login to their Yahoo accounts) and are redirected to their OpenID provider. After entering his credentials at the OpenID provider, the user is redirected to the login page. The login page checks to see if the user has been registered with the ASP.NET membership provider. If the user is registered he is redirected to the front page of the website. Unregistered users are redirected to the registration page where profile information is collected from the credentials returned by the OpenID provider (note that many providers don’t provide any information.) After registration, users may login and logout freely. The site captures a nickname for the user at registration (which may or may not be provided by the OpenID Provider) and greets the user with the nickname on the default page after login. The inspiration for my implementation was found [here](http://blog.madskristensen.dk/post/OpenID-implementation-in-Csharp-and-ASPNET.aspx).

# Performance Evaluation

The goal for OpenID is a seamless user experience. Many OpenID providers allow users to stay logged in indefinitely; thus users simply enter their OpenID identifier and are redirected to the content they wish to see. So in the way of actual performance there isn’t much to see. The real measure of performance on the site is what users don’t see. As such, the site does relatively well. The biggest problem was inconsistency in the support for profile information. Certain providers don’t provide anything at all, citing security concerns.

# Lessons Learned

Immediately after finishing my implementation, I found a second library at Google code that does exactly what my project does. I didn’t do any evaluation of that implementation but for future use I would prefer their library to my implementation. Another problem was with ASP.NET’s implementation of personalization – support for profiles in Visual Studio Web applications doesn’t work in the way one would expect. A lot of time was lost implementing a work around that cost valuable project time.

# Conclusions

As with any new technology, support is inconsistent; but a larger concern is splintering support by competing providers. Already, Facebook and Google are publishing OpenID libraries with similar names that are not interoperable. Also, four of the major providers (Google, Facebook, Yahoo, and Microsoft) claim OpenID support but don’t publish information on how to construct an OpenID identifier using your credentials from their system – To support their OpenID systems relying parties must implement proprietary technology from each provider. Also, security with OpenID is only as good as the OpenID provider being used. Yahoo is famously easy to breach, and most OpenID providers rely on simple username and password combinations for authentication. One can envision more secure OpenID implementations using two factor authentication or some other more secure technology but the cost associated with that would most likely be passed on to the user. I think people can have secure OpenID but only if they are willing to pay for it.

# References

Google. (2008, December). *DotNetOpenID*. Retrieved December 2008, from Google Code: http://code.google.com/p/dotnetopenid/

Kristensen, M. (2008, January 21). *OpenID implementation in Csharp and ASPNET*. Retrieved December 2008, from .NET Slave : http://blog.madskristensen.dk/post/OpenID-implementation-in-Csharp-and-ASPNET.aspx

MyVidoop. (2007, September 11). *OpenID Video*. Retrieved December 2008, from You Tube: http://www.youtube.com/watch?v=xcmY8Pk-qEk

*OpenID*. (2008, December). Retrieved December 2008, from Wikipedia: http://en.wikipedia.org/wiki/OpenID

*What is OpenID?* (2008, December). Retrieved December 2008, from OpenID Foundation: http://openid.net/what

1. [↑](#footnote-ref-2)