University of Idaho
CS481: Senior Design

Idaho Department of Health and Welfare

Time, Accounting, and Reporting System

prepared for

Don Moreaux
Marj Sanderson

and

The Idaho Department of Health and Welfare

Authors:
Scott Beddall
Brett Hitchcock
Chaylo Laurino
Alex Nilson

Advisors:
Greg Donohoe

December 7, 2011

Contents

1 Introduction

1.1 Identification

This Software Design Document pertains to the Idaho Department of Health and Welfare Time, Accounting, and Reporting System. Project development for Fall Semester, 2011 is executed by Scott Beddall, Brett Hitchcock, Chaylo Laurino, and Alex Nilson. The advisor for the project from the University of Idaho is Gregory Donohoe. The project sponsor and primary client from the Idaho Department of Health and Welfare is Don Moroeux.

1.2 Document Purpose, Scope, and Intended Audience

1.2.1 Document Purpose

This document's sole purpose is to outline the scope of Idaho TARS. This outline includes, but is not limited to:

- Development Decisions and Rationale.
- Architectural Specifications.
- Detailed Design Information.
- Locations of Other Project Resources.
- Complete Details Regarding Project Deliverables

1.2.2 Intended Audience for Document

Though the TARS is to be fully prototyped by the end of 2011, it will not be completed. With that being the case, this document is aimed at any future developers or users of Idaho TARS.

1.3 Software Purpose, Scope, and Intended Users

1.3.1 Software Purpose

Idaho TARS is intended to provide time and resource tracking for contractor/non-contractor work efforts within the Idaho Department of Welfare. Work efforts must be added to time-bounded project PCA codes and approved by users with sufficient privileges. Project summaries, cost totals, user logs, and other information will then be available within TARS to authorized users. These users will be authenticated by an Active Directory interface.

1.3.2 Software Scope/Context

The Idaho Department of Health and Welfare is currently utilizing a resource called Mariner for time management and accounting. The IDHW's needs, however, are significantly less than the capabilities that Mariner provides. Portfolio and Resource Management, Planning, and other features of Mariner are being paid for, but left unused. The under-utilization of Mariner, coupled with fact that the IDHW would prefer an open-source solution that meets their specific needs and workflow, motivated Don Moroaux to bring the TARS project to the University of Idaho.

1.3.3 Intended Users for the Software

Intended Users of Idaho TARS are the staff and employees of the Idaho Department of Health and Welfare as well as its contractors.

1.4 Definitions, Acronyms, and Abbreviations

MVC	Model View Controller. A design pattern used for content focused websites. Provides security through modularity, ease of maintenence, and a clear architecture.
TARS	Time, Accounting, and Reporting System.
PCA Code	Position Classification Allocation Code.
SQL	Structured Query Language. Used for input and retrieval of data from a SQL database.
IDHW	Idaho Deparment of Health and Welfare
Work Effort	A project. Has one or more assigned PCA Codes and a list of associated work tasks.
Connection String	A formatted line of text that contains all relevant connection information for a remote resource. (server, dsa)
LDAP	Lightweight Directory Access Protocol. A protocol used for interface with distributed directory services. (Active Directory, Apache Directory Server)
DSA	Directory Service Agent. A server that specifically listens for queries via LDAP.
Active Directory	Microsoft Directory Services
Apache Directory Services	Open source LDAP alternative to Active Directory. Highly stripped down.

1.5 Document Overview

Section 2 describes software constraints imposed by the operation environment, system requirements, and user characteristics. After this it will identify the system stakeholders and list/describe their concerns.

Section 3 of this document describes the system and software architecture from several viewpoints, including, but not limited to, the developer's view and the user's view.

Section 4 provides detailed design descriptions for every component defined in the architectural view(s).

Sections 5 provides traceability information connecting the original specifications (referenced above) to the architectural components and design entities identified in this document.

Section 6 and beyond are appendices including original information and communications used to create this document.

2 Software Requirements, Constraints, and User Characteristics

Check the requirements_summary.docx for an extensive list of requirements.

3 Software Architecture

3.1 Server Architecture - Microsoft Internet Information Services 7

One of the IDHW's requirements for the this project is the use of Windows Server infrastructure. In this case the TARS development team will use Microsoft IIS7. Microsoft's Internet Information Services server is a modular, intuitive server application. New considerations must now be applied, however. IIS7, being a Microsoft product, uses Microsoft development software. Namely:

- C#
- ASP.NET
- Visual Basic/VB.NET
- .NET Development Framework
- Active Directory Services

TARS will be developed using all these technologies, as well as the IIS7 Model-View-Controller application. The MVC application will be described in detail in the next section.

3.2 Model-View-Controller

Most of the heavy-lifting for TARS will be present in the display and interaction with large amounts of data. This problem is what the Model-View-Controller design pattern was created for. The idea is that each word in the acronym: Model, View, and Controller each represent a component that handles a different aspect of the display process.

Advantage of Model-View-Controller architecture:

- Separates the user interface from the logic used in the database.
- Allows for independant development, testing, and maintenance of these seperate parts of the application.

3.2.1 Controller

A Controller receives input from the user and converts it to instructions for the Model and View components. Most MVC's use the first argument after the website URL as the controller call. In our case:

```
http://idahotars.com/home
```

Will load the "home" controller default function (Index). The second entry in the url after the controller selection will select the specific function in the controller. Any subsequent entries will then make up arguments to the specific function.
Load the viewTimeSheet function in the user controller:

```
idahotars.com/user/viewTimeSheet
```

Load the viewTimeSheet function in the user controller with arguments 10 and 20:

```
idahotars.com/user/viewTimeSheet/10/20
```

This configuration can be viewed or modified in "global.aspx."

3.2.2 Model

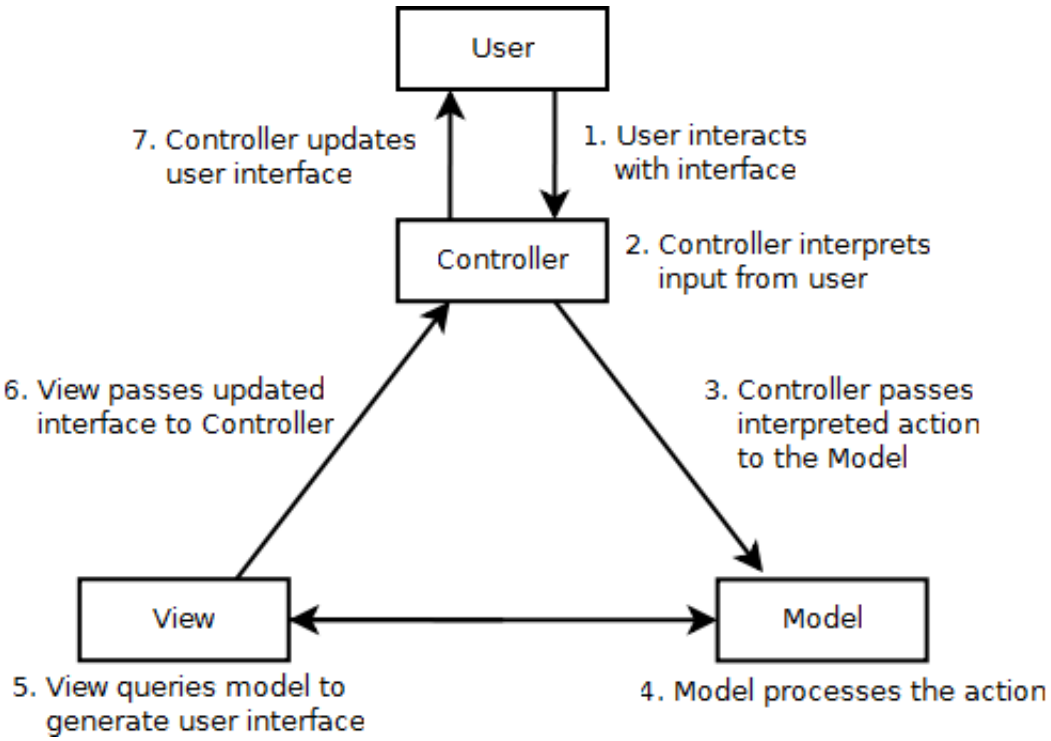
The Model manages database queries and assembles data for use elsewhere in the MVC. However, in the MVC3.0 extension specific to IIS7, there is a slight oddity when it comes to the Model component. That is, the Model doesn't actually execute any of the database interaction. Instead, the Models are used specifically as database table schemas, giving TARS a way to know what the remote tables look like. The actual interaction is carried out by Database Contexts. (DbContext class)

3.2.3 View

Views render the data received from the Models into viewable web pages.

3.3 SQL2008 Database

Though IIS can utilize any format of SQL Database, the IDHW requires that TARS use SQL2008. Any queries made by the MVC will be carried out by the "Model" component of the MVC.



3.4 Active Directory

The IDHW uses Microsoft Active Directory for their user authentification. With that infrastructure already in place, it is logical to use the same for Idaho TARS. To authenticate users of TARS, there will be three new Active Directory groups added: TARSAdmin, TARSManager, and TARSUser. If a given user is part of any of these three groups, they will be allowed access to TARS based upon their group.

To provide this functionality, the TARS development team wrote a helper class that can be used when an Active Directory connection is needed. This "LDAPConnection" class can be invoked within any controller, so long as "TARS.Helpers.LDAPConnection" is referenced.

Primary functions for now are:

```
LDAPConnection() //constructor, initiates variables needed for a connection
boolean requestUser(string user, string password); //returns true if the user/password combo exists
string requestRole(string user); //returns which group a user is part of
```

All these functions create an LDAP connection, query the DSA, and close the connection. The user of this helper class need not ever understand what is going on under the hood to use it correctly.

3.5 Security

While Idaho TARS will be used internally, there is still a security risk. The system may not be dealing with any highly confidential info, but TARS will have access to government resources like the IDHW Active Directory and I-Time interfaces. To prevent easy exploitation of the TARS databases, all queries to the database will be centralized in Model. This will will not only make the team's code simpler, but also make it more secure. Centralized queries allow us to easily adopt a strong security stance. In addition, TARS uses DbContexts for database interaction. With that being the case, no SQL is ever directly executed from TARS. This significantly reduces the risk of SQL Injection attacks.

3.6 Browser Interface

Though it is probably already indicated by the server architecture, the development team must make it clear that this software is being developed for a web interface. This will eliminate many of the dependencies that are inherent to a system launched from a binary. The development team is developing this project to meet the following end system requirements:

- 1024x768 monitor resolution
- Google Chrome
- Mozilla Firefox
- Internet Explorer 7 and up
- Safari
- Compliant with W3C standards

4 Design Descriptions

4.1 Model-View-Controller Modules

4.1.1 Global Scripts and Config Files

There are two extremely important configuration files within the Visual Studio solution. Unfortunately, two both named "Web.Config."

The first Web.Config is present in projectBase/Views/Web.Config. This config file handles various view configurations that affect TARS visually.

The second Web.Config is present in projectBase/Web.Config. It handles project level configuration; most importantly including the extremely important database connection strings.

The third, and by far the most important config file is projectBase/Global.aspx. This contains the functions that control routing for the entire MVC. This job includes determining routing to controllers, filtering requests, and registering the initial routes on a default request for idahotars.com.

4.1.2 Controllers

These controllers do not yet have a finalized set of associated functions and variables. When these are complete a full description will reside here.

- HomeController
- UserController

- ManagerController
- AdminController
- AccountController

The home controller is the default page in the case that a user is not logged in. It also provides the entry point to TARS when a controller is not specified.

AccountController provides login functionality. This includes calling the LDAPConnection helper class to authenticate users. AccountController was provided by a base example of an MVC, but is modified to serve the TARS specific needs.

UserController inherits from the default MVC Controller class as well as providing basic functionality for a normal user.

ManagerController inherits from UserController, but adds a couple more abilities that managers need as per the requirements specification.

AdminController inherits from ManagerController, inheriting all functionality as well as providing any and all administrative functions that are needed by TARS admin. Having these inherited privileges ensures that forced permission traversals will be almost impossible.

```
Home Controller //Default controller called when visited TARS
                //for the first time or when not logged in.

User Controller
 -> ManagerController
  -> AdminController
```

4.1.3 Models

- AccountModels (provided by IIS7 MVC)
- History
- Hours
- PcaCode
- PCAWE
- WorkEffort
- User

These items do not inherit from the default Model class. In addition, each class has an associated DbContext class in the same file. Example:

```
public class PCA_CodeDBContext : DbContext
{
   public DbSet<PcaCode> PCA_CodeList { get; set; }
}
```

These connections are not conventional classes or objects. To indicate this, their naming conventions are slightly more verbose. A DbContext creates a new context instance that utilizes an existing Database Connection String (located inside web.config) to create a connection to a SQL Database.

4.1.4 Views

- Every controller has an associated View. Located in their respective folders. Manager example:

```
Views/Manager/___.cshtml
   ///where ___ is a view associated with a given function within the controller
```

4.2 SQL2008 Database Schema and Interface Description

As mentioned above in the Software Architecture section, TARS will use a SQL2008 database to store all interactions other than User Info (handled by Active Directory). Before outlining the Database Schema, it would be wise to fully describe the thought process of the development team.

Stripped down to its most base parts, TARS is simply a database interface through which users can log and retrieve hours to and from work efforts. These "Work Efforts" are simply general projects that can have hours of contractor/non-contractor work added to their totals. For instance, there might be a Work Effort that is assigned its own unique ID and whos description is "Document the latest changes to the TARS SDD." Any employees who wish to log their hours will find the Work Effort's ID, add their hours along with other relevant data, and submit their entire timesheet for approval.

The Work Effort to "Document the latest changes to the TARS SDD." now has hours logged on it and waiting for approval. A user with the correct Active Directory permissions (part of group TARSManager or TARSAdmin) can now go check the status of the Work Effort, and approve any pending hours waiting on it. The development team has chosen to add all hours, approved or not, to the Work Effort's database table. A simple boolean present as a column in the table will ensure that filtering by approved/un-approved will be a simple task.

Unfortunately, the process is not done. Now that a Work Effort has hours charged to it, how can the accounting department charge these expenses? The simple answer is, they can't yet. To provide that functionality, PCA Codes must be assigned. This introduces another database table, as one PCA Code may have multiple Work Effort associations; just as one Work Effort may be associated with multiple PCA Codes.

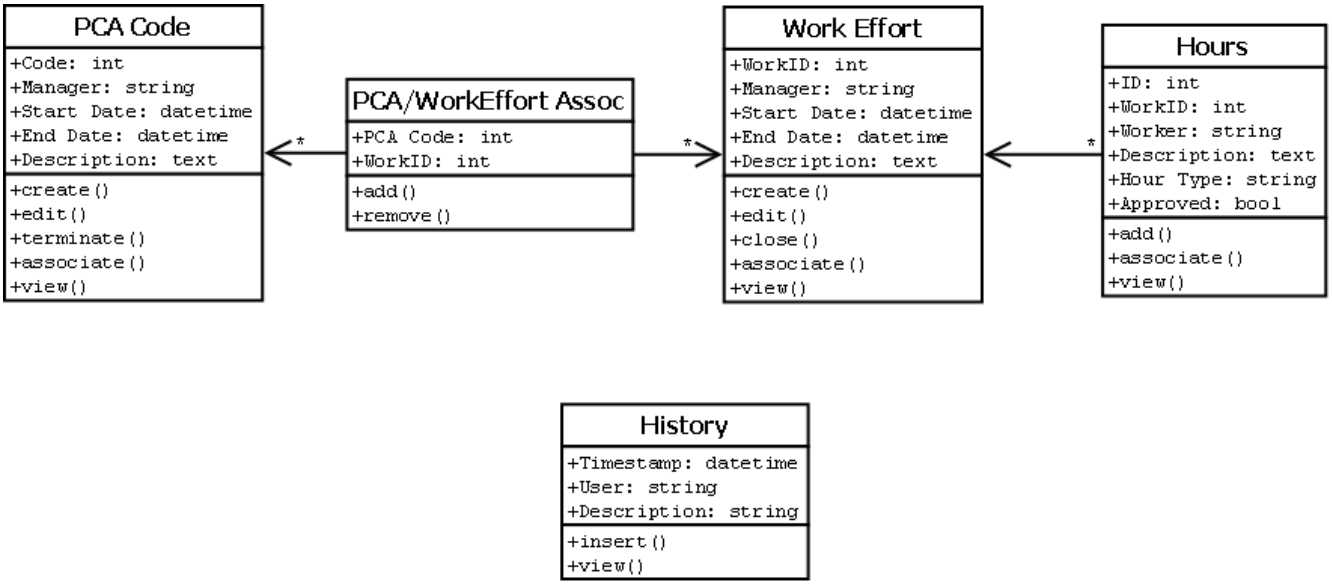
One final note about the Work Efforts and PCA Codes is needed. The requirements state that they both must be time-bounded, capable of early expiration, and renewable.

In addition, for each of these Models, there is also a modification to their paramaterized DbContexts. That is, DB.saveChanges() is overwritten to save a new entry in the History table before calling super.saveChanges().

Addendum for 11/1/2011. The stakeholders introduced the concept of "tasks." There will be a default set of tasks for every work effort, but members of the "administrator" Active Directory group will have the ability to add or remove these "tasks" from a given work effort. While the development team at one point believed it possible to create Tasks functionality without another database table, the belief was quickly proven infeasible. With that being the case, a new table was added.

Addendum as of 11/23/2011. The TARS development team has learned that while the Active Directory DSA will service user/password requests, TARS will be unable to push any changes to the Directory. With that being the case, the development team quickly deduced that yet another database table would be needed. Namely, the "User" table. This table will store unique userids as well as any web and role configuration information that may be needed by TARS in the future. For now, the development team is simply using it to store Role information. Later on, the configuration storage will be utilized by the next team.

4.2.1 TARS Database Schema



4.3 Naming Conventions

All defined classes and objects will use capitalized camelcase, with their first letter a capital as well.

Local Variables and instances of classes will use camelcase also, neglecting the capitalization of their first letter.

```
class Manager : Controller
{
    public int id;
    public bool approved = FALSE;
    public string hoursType;
}
```

5 Tracability Information

All effort on the project may be tracked via the project GitHub repository: github.com/ICBM/TARS

The project website resides at: http://seniordesign.engr.uidaho.edu/2011-2012/CostManagement/

Client requirements changes will be handled via email. Current progress on requirements as well as the requirements changelog is available at: github.com/ICBM/TARS/blob/master/doc/requirements_summary.docx

6 Deliverables

6.1 Deliverables Overview

At the end of the first semester the TARS team turned over the following deliverables to be passed on to the next semester's team for completion; This Design Document, Client Requirement Documents, Meeting

Minutes, Tutorials, Prototype Source Files, Requirements Summary, and our GitHub Repository. For exact locations, check the "GitHub Locations of Importantance Section"

This Design Document contains all of the conceptual and technical designs for this project. It also includes various Client restrictions on software and platform for the project. For further details please read the document.

The Client Requirement Documents outline the Client's desired functionality of the system. It is broken down into 6 parts: PCA, Data, Reporting, View, Security, Navigation, and Workflow. This document was provided to us by the Client. It has been updated several times as requirements have been added or altered.

All of the Minutes of our meetings with our clients as also been added. They show the changes in the requirements over time and contain clarifications to questions we or the client may have had during the course of the project.

Extensive trial and error was involved in setting up our machines to be viable development platforms based on the restrictions laid out by the Client. To prevent our successors from having to deal with the same problems we dealt with we have created several tutorials to help smooth the process for them.

We have managed to produce a functioning prototype. It includes all the primary models and controllers but has only rudimentary views and testing. The source code has been provided as has a Microsoft Visual Studio build. The full details concerning the functional state of the prototype in relation to the full project requirements are laid out in the Requirements Summary document.

The Requirements Summary is a combination of the Client Requirement Document and a full summary of our prototypes functionality. It explains in detail the current status of every requirement.

The final deliverable we have is our GitHub Repository. It has been the primary location for our collaborative work. It is an open source repository as the client desired us to make the source code public. It costs 12 dollars a month and will need to be transferred to the new team as soon as they take up the project.

6.2 Prototype Deliverable Detail: Current Project Status

6.2.1 Project Architecture

6.3 GitHub Locations of Importantance

The root directory is present at github.com/ICBM/TARS

- SSD: /doc/ssd.tex – /doc/ssd.pdf
- Meeting Minutes: /minutes
- Tutorials: /doc/Tutorials-Resources/
- Prototype: /TARS
- Requirements Summary: /doc/requirements.docx
- Client Requirements Document: /doc
- Contact Information: /contact_info.pdf

- SQL Config Dumps: /doc/Tutorials-Resouces/SQL/
- LDAP Config Dumps: /doc/Tutorials-Resouces/LDAP/

7 Appendix A: Use Cases

Login

1. Click the "Login" button
2. Enter username and password
3. Hit Enter or click "Submit"
4. System authenticates and redirects to home page.

Adding new PCA code.

1. Select "Add PCA"
2. System loads PCA form
3. Fill in form, including time bounds for the PCA code
4. Press "Submit"
5. System updates tables and redirects to new PCA display page
6. A request to the financial department will be automatically dispatched. PCA codes are not actually generated by TARS.

Deactivating PCA code

1. Select desired PCA code
2. System loads PCA display page
3. Click deactivate
4. Confirm deactivation.
5. System updates tables and locks PCA code.

Adding Work Effort

1. Select "Add Work Effort"
2. System loads Work Effort form
3. Fill in form
4. Associate desired PCA code or codes, if multiple PCA codes are chosen a percentage of work effort may be set for each
5. Associate entity or entities
6. Click "Submit"
7. System updates tables and redirects

Updatinging Work Effort

1. Select "Update Work Effort"
2. System loads Work Effort form
3. Make Changes in form
4. Click "Submit"
5. System updates tables and redirects

Adding Hours

1. Select "Add Hours"
2. System loads Work Effort Selection form
3. Select Work Effort
4. System loads Hours form
5. Fill in form or select "replicate" to fill with previous weeks data
6. Click "Submit"
7. System updates tables and redirects

Approving Hours

1. Select "Approve Hours"
2. System loads hour approval form and fills with items needing approval
3. Select item
4. Select "Approve" or "Disapprove"
5. System updates tables and redirects back to hour approval form

Adding Tasks

1. Select "Add Tasks"
2. System loads Work Effort Selection form
3. Select Work Effort
4. System loads Task List form
5. Add desired tasks
6. Click "Submit"
7. System updates tables and redirects

Edit/Update Employee/Contractor Data

1. Select desired entity
2. System pulls data from database and loads Entity Form
3. Edit/Update as desired
4. Click "Update"
5. System updates table and redirects.

View History

1. Select "View History"
2. System loads History Search form
3. Enter Search Criteria
4. System loads Search Result form and fill with data

The system must allow for future time entry

1. Select "Add Work Effort"
2. Fill in form, making sure to add the effort to the correct date.
3. Click "Submit"

All data for reporting shall be extracted via external source (EDW. Excel, etc.).

1. Under a given PCA code or work effort, select "Get Data Report".
2. The system will then generate a copy of the data in EDW or Excel format.
3. User saves the copy at a destination of their choosing.

Must allow users to create a view of their I-Time timesheet.

1. User logs in.
2. On the user's personal page, select "Get I-Time Report".
3. The system will then generate a copy of the data for viewing.

Must have a sort and group function that allows work effort to be grouped by application, division, manager, etc.

1. User logs in.
2. User selects "My Work Efforts"
3. System generates list of all work efforts related.
4. User selects to sort by name, application, etc.
5. System sorts and redisplays work efforts in the proper order.

The system must allow a user the ability to create a custom view of the data.

1. User selects a PCA code or Work Effort code
2. User selects "Get Data Report".
3. User enters custom settings and hits "Select".
4. System generates report.

Must allow users to easily size windows

1. User resizes browser, which will resize the web interface.

The system shall provide search/find functionality to locate work efforts, with minimal amount of navigation (less than 5 clicks per important action)

1. User logs in.
2. User navigates to a work effort by...
    - Clicking on a link in the "My Recent Work" bar.
    - Entering a Work Effort code or PCA code in the Search bar on the top.

8 Appendix C: Reference Links

Unit Testing ASP.NET MVC
http://msdn.microsoft.com/en-us/magazine/dd942838.aspx

SQL2008 integration w/ IIS7 MVC reference
http://blog.evonet.com.au/post/Setting-up-SQL-Server-2008-for-an-ASPNET-website-on-IIS-70.aspx

SQL2008 St15udio Manager
http://www.microsoft.com/download/en/details.aspx?id=7593

SQL2008 Server
http://www.microsoft.com/download/en/details.aspx?id=1695

Apache Directory Server (LDAP stand-in for Active Directory)
http://directory.apache.org/apacheds/1.5/

Apache Directory Studio (Essential Client for Managing ApacheDS)
http://directory.apache.org/studio/

Write an LDAP interface in C#
http://www.youcanlearnseries.com/programming%20Tips/CSharp/LDAPReader.aspx