USB를 이용한 Smart Class

|  |
| --- |
| 요즈음 학교 수업의 환경은 점차 전자화 되어가고 있다. 실제로 몇몇 기관은 컴퓨터 및 전자기기로만 수업을 진행하는 곳도 있다. 이러한 추세에 따라 제시되어있는 현재의 솔루션들은 기본 전자기기가 테블릿 pc, 전자칠판 등 가격이 비싼 장비들을 기반으로 제시되어 있다. 그리고 수업에 필요한 기본 교재 자체도 일반 시중에 판매되는 종이로 된 책을 사용하고 있어 경제적이지 못하다.  일단, 본 프로그램의 필요한 전자기기로는 pc와 usb로, 모든 수업과 작업이 이루어진다. 그리고 수업에 필요한 교재들을 이북화 하여 pdf파일로 수업을 진행한다. 이러한 점들이 복합적으로 전체적으로 교육 솔루션에 필요한 가격 절감성을 유발한다.  또한, usb내의 자료들을 기반으로 바탕화면을 제공해 기존의 바탕화면처럼 쓸 수 있고 따로 자주 사용하는 유틸 등록 및 데이터 공유 등 편의성을 위한 다양한 기능을 제공한다. 본 프로그램은 가격 절감성, 휴대성, 편의성을 착안하여 제작되었다. |

목 차

1절 프로젝트 개요

* 프로젝트 명………………………………………… 3
* 서론 ………………………………………………… 3
* 프로젝트 개방 목적 ……………………………… 3
* 팀원구성 …………………………………………… 4
* 개발 일정…………………………………………… 4
* 부분별 수행업무 및 담당자……………………… 5
* 프로젝트 개발 범위 ……………………………… 5

2절 배경지식

2.1 WPF…………………………………………………… 5

* WCF…………………………………………………… 7

3절 시스템 구축 및 개발 내용

3.1 시스템 ………………………………………………… 8

1) U.S.C 기능 ………………………………………… 9

4절 결론 및 개선 방안

5.1 기대 효과 ………………………………………… 31

5.2 개선 방안 ………………………………………… 31

5절 참고문헌

U.S.C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 성명 | 학교 | 학번 | 전공 | 전자메일 |
| 최영 | 우송대학교 | 200910487 | IT경영정보학과 | Cy2005@naver.com |
| 정우창 | 우송대학교 | 200810664 | IT경영정보학과 | inri501@naver.com |
| 성연상 | 우송대학교 | 200810641 | IT경영정보학과 | dustkdwkd@naver.com |
| 박광열 | 우송대학교 | 200711455 | 글로벌비지니스학부 일본어전공 | gawallsibya@naver.com |

1절 프로젝트 개요

* **프로젝트 명**

U.S.C(교육솔루션)

* **서론**

현재 스마트 스쿨 시장은 점차 그 적용 범위를 넓혀갑니다. 하지만 현재제시된 솔루션은 공용 pc상에서 이미 수업을 진행하고 있는 학교에서의 그 적용이 불필요한 비용부담을 유발시킨다. 하여 공용 pc상에서의 가으이 솔루션을 제시하여, 이를 통해 효과적인 교육을 진행코자 한다.

* **프로젝트 개발 목적**

기본적으로 공용 pc상에서 이루어지는 수업을 위한 솔루션이기 때문에기존의 공용pc의 단점을 보자면 여러 사람이 같이 사용하기 때문에 각자의 폴더나 작업내용이 중복될 여지가 많다. 예를 들어, 어제 내가 작업한 내용이 오늘 다른사람의 작업 내용과 겹치거나 폴더를 옮겼을 경우 자신의 파일을 찾는 것에 난해함을 겪을 것이다. 이에 기존의 바탕화면과 똑같은 하지만 개별적으로 운용 가능한 개인 바탕화면을 제공함으로써 편의성을 증대시키고자 한다. 또한, 기존 솔루션들의 비용 부담인 DB의 사용을 XML문서를 통한 정보 저장으로 라이선스 비용을 최소화한다.

* **팀원 구성**

팀 장 : 최영

팀 원 : 정우창, 성연상, 박광열

* **개발 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 기 간  업 무 | | 5월 | | 6월 | | | | 7월 | | | | | 8월 | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 회 의 | 주제 선정 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 설 계 | 요구/분석 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 자료 조사 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 설 계 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 프로그래밍 | USB 정보 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 필기정보 저장 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 서버 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 개인 바탕화면 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| PDF Viewer |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 개인 노트 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 동기화 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 데이터 공유 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 테스팅/디버깅 | 프로젝트 테스팅 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 디버깅 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

* **부분별 수행업무 및 담당자**

|  |  |  |
| --- | --- | --- |
| 담당업무 | 사용툴 | 담당자 |
| 통합 및 아키텍쳐 구상 | C#,WCF | 최영, 정우창 |
| 서버, 클라이언트 | C#,WCF | 최영, 정우창, 성연상 |
| DB | XML | 최영, 정우창 |
| 파일 전송, 다운로드 | C#,WCF | 성연상, 박광열 |
| UI 설계 | C#,WPF | 정우창 |
| UI 디자인 제작 | C#,WPF | 정우창, 성연상 |
| USB 정보 | C# | 박광열 |
| PDF Viewer | C#,WPF | 최영 |
| 개인 바탕화면 | C#,WPF | 정우창 |

* **프로젝트 개발 범위**
* WPF를 이용해 깔끔하고 사용하기 쉬운 UI
* usb 인식을 기반한 바탕화면 출력
* 여러 문서를 보여주는 통일된 뷰어
* 과제 제출 및 문서 다운로드를 위한 데이터 공유

**2절 배경지식**

**2. 1 WPF(Windows Presentation Foundation)**

Windows Presentation Foundation 곧 WPF는 아발론(Avalon)이라는 코드 이름을 사용하였으며 닷넷 프레임워크 3.0의 그래픽 하부 시스템 기능(WinFX)이며, XAML과 직접 관련이 있다. 마이크로소프트 윈도 운영 체제의 최신 버전인 윈도 비스타의 경우 미리 설치가 되어 있다. WPF는 윈도 XP 서비스팩 2와 윈도 서버 2003에서 설치 프로그램을 내려 받아 사용할 수 있다. 응용 프로그램을 만들기 위한 일정한 프로그래밍 모델을 제공하며 사용자 인터페이스와 비즈니스 로직를 분명히 구분한다. WPF 응용 프로그램은 바탕 화면에 배치하거나 웹 브라우저에 호스팅할 수 있다. 또한 리치 컨트롤, 설계, 윈도 프로그램의 시각 항목을 사용할 수 있다.

**Windows Presentation Foundation**

WPF의 목표는 다음과 같은 응용 프로그램의 서비스의 호스트를 통일하는 것이다:

● 사용자 인터페이스

● 2D 및 3D 그리기

● 고정, 적합 문서

● 고급 타이포그래피

● 벡터 그래픽스

● 레스터 그래픽스

● 애니메이션

● 데이터 바인딩

● 오디오

● 비디오

마이크로소프트 실버라이트는 웹 기반의 WPF 부분 집합이다. 개발하는 동안에는 "윈도 프레젠테이션 파운데이션/에브리웨어"(Windows Presentation Foundation/Everywhere)의 준말 WPF/E라는 이름이 붙여졌다. 실버라이트 부분 집합은 플래시와 같은 웹과 휴대 응용 프로그램을 윈도 닷넷 응용 프로그램과 같은 코드를 가지고도 사용할 수 있게 한다. 3차원 기능은 지원하지 않지만 XPS, 벡터 기반의 그리기가 포함되어 있다.

**도구**

윈도 프레젠테이션 파운데이션 응용 프로그램을 개발하는 데 이용할 수 있는 개발 도구가 여러 개 있다. 그 가운데 대부분이 베타 품질이며 일부는 상용화되어 있다.

● 마이크로소프트 사이더: 비주얼 스튜디오 2005 IDE를 위한 추가 기능 형태로 짤 수 있는 XAML 디자이너로, 윈도 파운데이션 프레젠테이션 응용 프로그램을 작성할 수 있다. 비주얼 스튜디오 2005로부터 CTP 형태로 이용할 수 있어서 비주얼 스튜디오 2005에 RTM 버전에 대한 계획은 없다. 사이더는 비주얼 스튜디오 2008에 통합되어 있다.[1]

● 마이크로소프트 익스프레션 블렌드: 2차원/3차원 그래픽스, 텍스트, 폼 콘텐츠가 포함된 WPF 응용 프로그램을 작성하기 위한 캔버스를 제공하는 디자이너 지향 도구이다. 다른 도구에 추가할 수 있는 XAML을 만들어 낸다.

● 마이크로소프트 익스프레션 디자인: XAML에 내보낼 수 있는 비트맵, 2차원 벡터 그래픽스 도구이다.

**WPF 응용 프로그램**

WPF는 데스크톱의 윈도 응용 프로그램을 위한 차세대 그래픽 API로 고안되어 있다. 다음에는 WPF로 작성된 마이크로소프트 자사 제품 중 일부 또는 다른 서드 파티 회사의 주류 응용 프로그램을 가리킨다.

●마이크로소프트 익스프레션 디자인

●마이크로소프트 익스프레션 블렌드

●야후! 메신저

●노키아 뮤직 PC 베타

●클리어 오피스 스프레드시트

●MuvUnder Cover: The Album Art Sleuth

**2.2 WCF**

윈도 커뮤니케이션 파운데이션(Windows Communication Foundation, WCF)은 응용 프로그램을 한 대의 컴퓨터나 네트워크에 연결된 여러 컴퓨터를 걸쳐 통신할 수 있게 만들어 주는 새로운 통신 하부 시스템이다. WCF 응용 프로그램들은 닷넷 런타임을 대상으로 하는 언어를 사용하여 개발할 수 있다.

닷넷 프레임워크 3.0의 일부로 도입된 4 개의 API 중 하나이며 윈도 비스타, 윈도 서버 2008에 포함되어 있다. 윈도 XP, 윈도 서버 2003에서도 지원된다.

WCF 프로그래밍 모델은 웹 서비스, 닷넷 리모팅, MTS, 그리고 메시지 쿼리를 하나의 서비스 지향 프로그래밍으로 통합한다. 로컬 컴퓨터, 랜, 인터넷 안의 프로세스 통신어를 위한 단일 API와 더불어 웹 서비스 개발에 대한 고속 개발 도구 방식을 제공하기 위해 고안된 것이다. WCF는 모래상자(sandbox) 안에서 실행되며 모든 닷넷 응용 프로그램이 제공하는 강화된 보안 모델을 제공한다.

WCF는 두 개의 프로세스 사이의 통신을 위해 SOAP 메시지를 사용한다. 그럼으로써 WCF 기반 응용 프로그램을 SOAP 메시지를 통해 통신하는 다른 프로세스와 상호 운영할 수 있게 만들어 준다. WCF 프로세스가 WCF가 아닌 프로세스와 통신할 때, XML 기반 인코딩은 SOAP 메시지를 위해 사용되지만, 다른 WCF 프로세스와 통신할 때에는 SOAP 메시지는 최적화된 이진 포맷으로 인코딩된다. SOAP 포맷의 데이터 구조에 순응하는 인코딩들 둘 다 정보 세트라고 부른다.

서비스 지향 아키텍처WCF는 서비스 지향 아키텍처 원칙과 부합하도록 설계되어 있으며 서비스가 소비자에 의해 소비되는 분산 컴퓨팅을 지원한다. 클라이언트들은 여러 개의 서비스를 소비하며 서비스들은 여러 대의 클라이언트에 의해 소비될 수 있다. 서비스들은 보통 WCF 클라이언트가 서비스 소비에 사용할 수 있는 WSDL 인터페이스를 가지고 있다. WCF는 WS 어드레싱, WS 신뢰 메시지, WS 보안과 같은 수많은 고급 웹 서비스 (WS)를 사용한다.

**3절 시스템 구축 및 개발 내용**

**3.1 시스템**



**[프로그램의 아키텍쳐]**

**- Client : 사용자가 사용하는 프로그램.**

1. UI Module : 사용자가 UI를 통하여 기능을 사용할 수 있게 됨.

2. CMember: Member Add, Member Login의 기능을 관리한다.

2.1 Member Add : 사용자 등록

2.2 Member Login : 사용자 로그인

3. CWallPaper: WallPaper Info, WallPaper Change의 기능을 관리한다.

3.1 WallPaper Info: 전체적인 바탕화면 정보 및 데이터 공유

3.2 WallPaper Change: 바탕화면 설정

4. CViewer: Auto Save, Upload, Edit의 기능을 관리한다

4.1 Auto Save : 자동저장 기능

4.2 Upload : 업로드기능

4.3 Edit : 문서 필기 기능

**- Server : 사용자가 사용하는 자료, 데이터를 보관하는장소**

1. Manager : 클라이언트로부터 오는 데이터를 필터링하게됨

2. Member : 사용자에 관련된 부분을 관리

3. WallPaper : WallPaper에 관한 부분을 관리

4. Viewer: Viewer에 관한 부분을 관리

5. DataBase : 데이터들의 입, 출력을 관리하는 부분

6. User Folder : 사용자의 파일을 실질적으로 관리하는 부분

**3.1.1 U.S.C 기능**

* 로그인

- usb 삽입시 로그인

- 서비스에서 명령행 인수로 시리얼 넘버 전달

|  |
| --- |
| private void Application\_Startup(object sender, StartupEventArgs e)  {  Service service = new Service();  GetUsbPath getusbpath = new GetUsbPath();  Application.Current.Properties["USBDrivePath"] = getusbpath.GetUsbDriverName();  if (e.Args.Length > 0)  {  string serial = e.Args[0];  if (service.Find(serial))  {  Application.Current.Properties["Members"] = service.Join(serial);  if ((string[])Application.Current.Properties["Members"] == null)  {  try  {  Application.Current.Properties["Members"] = service.Join(serial);  }  catch  {  return;  }  }  Application.Current.Properties["User"] = serial;  string userinfo = service.FindUserInfo(serial);  WindowsStyle window = new WindowsStyle(service, userinfo);  window.Show();  }  else  {  USB\_Join usb\_window = new USB\_Join(service, serial);  //usb\_window.Icon = new BitmapImage(new Uri(@"../../아이콘/HP-Flash-Drive.ico", UriKind.Relative));  usb\_window.Show();  }  }  } |

* 바탕화면
* 프로그램 시작시 바탕화면 사이드에서 프로그램을 구동함
* 인자로는 메인 윈도우 객체를 전달한다.

|  |
| --- |
| * [DllImport("user32.dll")] * private static extern IntPtr FindWindow(string lpClassName, string lpWindowName); * [DllImport("user32.dll")] * private static extern IntPtr GetWindow(IntPtr hWnd, int uCmd); * [DllImport("User32.dll")] * private static extern IntPtr SetParent(IntPtr hWndChild, IntPtr hWndNewParent); * private const int GW\_CHILD = 5; * public static void SetOnDesktop(Window window) * { * IntPtr Progmanhwnd = IntPtr.Zero; * IntPtr hwd1, hwd2, hwd3; * Progmanhwnd = (IntPtr)FindWindow("Progman", "Program Manager"); * hwd1 = GetWindow(Progmanhwnd, GW\_CHILD); * hwd2 = GetWindow(hwd1, GW\_CHILD); * hwd3 = GetWindow(hwd2, GW\_CHILD); * SetParent(new WindowInteropHelper(window).Handle, hwd2); * } |

* 파일 업로드
* 프로그레스 바 Show

|  |
| --- |
| private void UploadButton\_Click(object sender, EventArgs e)  {  OpenFileDialog dlg = new OpenFileDialog()  {  Title = "Select a file to upload",  RestoreDirectory = true,  CheckFileExists = true  };  if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  {  path = dlg.FileName;  if (bw.IsBusy != true)  {  // Start the asynchronous operation.  bw.RunWorkerAsync();  }  }   * } |
| ProgressBar pb;  string path = null;  private void backgroundWorker1\_DoWork(object sender, DoWorkEventArgs e)  {  Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal, new Action(delegate  {  pb = new ProgressBar();  pb.Show();  }));  FileInfo fileInfo = new FileInfo(path);  if (fileInfo.Exists)  {  using (FileStream stream = new FileStream(path, System.IO.FileMode.Open, System.IO.FileAccess.Read))  {  using (StreamWithProgress uploadStreamWithProgress = new StreamWithProgress(stream))  {  uploadStreamWithProgress.ProgressChanged += uploadStreamWithProgress\_ProgressChanged;  service.UploadFile(fileInfo.Name, fileInfo.Length, 0, uploadStreamWithProgress, flag);  }  }  }   * } |
| private void backgroundWorker1\_ProgressChanged(object sender, ProgressChangedEventArgs e)  {  this.pb.Value = e.ProgressPercentage;  }  public void uploadStreamWithProgress\_ProgressChanged(object sender, StreamWithProgress.ProgressChangedEventArgs e)  {  if (e.Length != 0)  {  bw.ReportProgress((int)(e.BytesRead \* 100 / e.Length));  }   * } |

* 파일 다운로드
* 기본 디렉토리 생성

|  |
| --- |
| private void DownloadFile(string[] filepath)  {  string npath = "";  if (flag == 2)  {  npath = @"SchoolInteligentUSB\FileRoom";  CreateFolder(npath);  }  else  {  npath = @"SchoolInteligentUSB\BookRoom";  CreateFolder(npath);  }  System.Windows.Forms.FolderBrowserDialog dlg = new System.Windows.Forms.FolderBrowserDialog()  {  ShowNewFolderButton = true,  Description =  "Select the directory that you want to use as the default.",  SelectedPath = System.IO.Path.Combine(usbpath, npath),  // RootFolder = new DirectoryInfo(System.IO.Path.Combine(usbpath, npath)),    };  dlg.ShowDialog();  //프로그래스바 ㄱㄱ  MultiProgressBar mbar = new MultiProgressBar(service, filepath, flag, dlg.SelectedPath);  mbar.getfile += GetFile;  mbar.Show();   * } |
| void bw\_DoWork(object sender, DoWorkEventArgs e)  {  string usbpath = (string)System.Windows.Application.Current.Properties["USBDrivePath"];  if (pathflag == 1)  {  DirectoryInfo di = new DirectoryInfo(usbpath + @"SchoolInteligentUSB\BookRoom");  if (di.Exists != true)  {  di.Create();  }  }  else  {  DirectoryInfo di = new DirectoryInfo(usbpath + @"SchoolInteligentUSB\FileRoom");  if (di.Exists != true)  {  di.Create();  }  }  //--------------------------------------------------------------------------------------- 요거  copydirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + "\\" + "Temp";  DirectoryInfo ndi = new DirectoryInfo(Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + "\\" + "Temp"); //c:에 저장할 폴더 임시로 만듬  if (ndi.Exists != true)  {  ndi.Create();  }  //----------------------------------------------------------------------------------------  string fileName = "";  Stream inputStream;  long startlength = 0;  // long totalsize = service.GetTotalSize(pathflag);  long length = 0; // 한 파일 사이즈를 최종 사이즈까지 더한다.  while (totalsize > length)  {  long original = 0;  if (nowdownloadflag == 0)  {  original = service.DownloadFile(idx++, ref fileName, ref startlength, out inputStream, pathflag); //한파일 사이즈  }  else  {  original = service.DownloadFiles(filepath, idx++, pathflag, ref fileName, ref startlength, out inputStream); //한파일 사이즈  }  length += original;  string tempdirectroy = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments) + "\\" + "Temp";  using (FileStream writeStream = new System.IO.FileStream(tempdirectroy + "\\" + Path.GetFileName(fileName), FileMode.OpenOrCreate, FileAccess.Write))  {  int chunkSize = 2048;  byte[] buffer = new byte[chunkSize];  do  {  int bytesRead = inputStream.Read(buffer, 0, chunkSize);  if (bytesRead == 0) break;  writeStream.Write(buffer, 0, bytesRead);  Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal, new Action(delegate  {  //this.pb.Value = (int)(writeStream.Position \* 100 / totalsize);  this.pb.Value = (int)(writeStream.Position \* 100 / original);  this.total.Value = (int)(length \* 100 / totalsize);  }));  }  while (true);  writeStream.Close();  }  inputStream.Dispose();  }  } |

* 뷰어

- pdf 파일 로드

|  |
| --- |
| private void GetPDFView(string path, string page)  {  pdf.Children.Clear();  System.Windows.Controls.Image image = new System.Windows.Controls.Image  {  Source = Imaging.CreateBitmapSourceFromHBitmap(ExtractPage(path, int.Parse(first.Text)).GetHbitmap(), IntPtr.Zero, Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions())  };  pdf.Children.Add(image);  } |
| public static Bitmap ExtractPage(string pdfFilename, int pageNumber)  {  var pageNumberIndex = Math.Max(0, pageNumber - 1); // pages start at index 0  using (var stream = new PdfFileStream(pdfFilename))  {  IntPtr p = NativeMethods.LoadPage(stream.Document, pageNumberIndex); // loads the page  var bmp = RenderPage(stream.Context, stream.Document, p);  NativeMethods.FreePage(stream.Document, p); // releases the resources consumed by the page  return bmp;  }  } |
| static Bitmap RenderPage(IntPtr context, IntPtr document, IntPtr page)  {  Rectangle pageBound = NativeMethods.BoundPage(document, page);  float zoomFactor = size;  Matrix ctm = new Matrix();  IntPtr pix = IntPtr.Zero;  IntPtr dev = IntPtr.Zero;  var currentDpi = DpiHelper.GetCurrentDpi();  var zoomX = zoomFactor \* (currentDpi.HorizontalDpi / DpiHelper.DEFAULT\_DPI);  var zoomY = zoomFactor \* (currentDpi.VerticalDpi / DpiHelper.DEFAULT\_DPI);  // gets the size of the page and multiplies it with zoom factors  int width = (int)(pageBound.Width \* zoomX);  int height = (int)(pageBound.Height \* zoomY);  mycan.Width = width;  mycan.Height = height;  // sets the matrix as a scaling matrix (zoomX,0,0,zoomY,0,0)  ctm.A = zoomX;  ctm.D = zoomY;  // creates a pixmap the same size as the width and height of the page  pix = NativeMethods.NewPixmap(context, NativeMethods.FindDeviceColorSpace(context, "DeviceRGB"), width, height);  // sets white color as the background color of the pixmap  NativeMethods.ClearPixmap(context, pix, 0xFF);  // creates a drawing device  dev = NativeMethods.NewDrawDevice(context, pix);  // draws the page on the device created from the pixmap  NativeMethods.RunPage(document, page, dev, ctm, IntPtr.Zero);  NativeMethods.FreeDevice(dev); // frees the resources consumed by the device  dev = IntPtr.Zero;  // creates a colorful bitmap of the same size of the pixmap  Bitmap bmp = new Bitmap(width, height, System.Drawing.Imaging.PixelFormat.Format24bppRgb);  var imageData = bmp.LockBits(new System.Drawing.Rectangle(0, 0, width, height), ImageLockMode.ReadWrite, bmp.PixelFormat);  unsafe  { // converts the pixmap data to Bitmap data  byte\* ptrSrc = (byte\*)NativeMethods.GetSamples(context, pix); // gets the rendered data from the pixmap  byte\* ptrDest = (byte\*)imageData.Scan0;  for (int y = 0; y < height; y++)  {  byte\* pl = ptrDest;  byte\* sl = ptrSrc;  for (int x = 0; x < width; x++)  {  //Swap these here instead of in MuPDF because most pdf images will be rgb or cmyk.  //Since we are going through the pixels one by one anyway swap here to save a conversion from rgb to bgr.  pl[2] = sl[0]; //b-r  pl[1] = sl[1]; //g-g  pl[0] = sl[2]; //r-b  //sl[3] is the alpha channel, we will skip it here  pl += 3;  sl += 4;  }  ptrDest += imageData.Stride;  ptrSrc += width \* 4;  }  }  bmp.UnlockBits(imageData);  NativeMethods.DropPixmap(context, pix);  bmp.SetResolution(currentDpi.HorizontalDpi, currentDpi.VerticalDpi);  return bmp;  } |

* 시간표 설정

|  |
| --- |
| * private void Mon\_MouseDown(object sender, MouseButtonEventArgs e) * { * var element = (UIElement)e.Source; * if (e.ClickCount == 2 && element is Grid) * { * Grid Day = sender as Grid; * Setting set = new Setting(); * set.SetData(Day.Name, TimeRow); * if (set.ShowDialog() == true) * { * string start = set.StartTime; * string end = set.EndTime; * int startidx = TimeRow.IndexOf(start); * int endidx = TimeRow.IndexOf(end) - startidx; * CreateRowDefinition(Day); * Label l = new Label(); * l.Style = (Style)this.Resources["CommonLabel"]; * TextBlock tb = new TextBlock(); * tb.TextWrapping = TextWrapping.WrapWithOverflow; * tb.Text = set.Sub; * tb.Margin = new Thickness(0, 0, 0, 0); * l.Content = tb; * l.SetValue(Grid.RowProperty, startidx); * l.SetValue(Grid.RowSpanProperty, endidx+1); * l.ContextMenu = (ContextMenu)this.Resources["context"]; * if (!CheckReCreate(Day, startidx, endidx)) * return; * Day.Children.Add(l); * TimeXml.Save(Day.Name,set.Sub,start,end,set.Book); * } * } * } |

- 시간표 설정에 따른 교재 출력

|  |
| --- |
| private void TimeBookLoad(ArrayList date)  {  if (date != null)  {  foreach (string[] time in date)  {  string[] start = time[1].Split('~');  string[] end = time[2].Split('~');  DateTime dt = Convert.ToDateTime(start[0]);  if (Convert.ToDateTime(start[0]) < DateTime.Now && DateTime.Now < Convert.ToDateTime(end[end.Count<string>()-1]))  {  string book = time[0];  FileOpen(book);  }  }  }   * } |

* 필기 정보 xml

|  |
| --- |
| * class DrawXml * { * static private XmlDocument xmlDoc; * static private XmlElement DrawElement; * static private string DrawSave = null; * static private string usbpath; * static public MainWindow PdfWin * { * get; * set; * } * static public void BookName(string book) * { * usbpath = (string)Application.Current.Properties["USBDrivePath"]; * string name = System.IO.Path.GetFileNameWithoutExtension(book); * string ext = System.IO.Path.GetExtension(book); * DrawSave = usbpath + @"SchoolInteligentUSB\UserWindowXml\" + name + "(" + ext + ").xml"; * } * static public void SavePath(ref string path) * { * xmlDoc = null; * path += DrawSave; * } * static private void Init() * { * xmlDoc = new XmlDocument(); * try * { * xmlDoc.Load(DrawSave); * XmlNodeList asd = xmlDoc.GetElementsByTagName("Draw"); * DrawElement = (XmlElement)asd[0]; * } * catch * { * DrawElement = xmlDoc.CreateElement("Draw"); * } * } * static private XmlNode FindTypeNode(string page, string type, int strokeidx) * { * XmlNodeList idxlist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']/Type[@Value='" + type + "']/Index[@Value='" + strokeidx.ToString() + "']"); * if (idxlist[0] != null) * { * XmlNode pointnode = idxlist[0].LastChild; * if (pointnode != null) * return pointnode; * } * return null; * } * static public void Xml\_Save(string type,int strokeidx, System.Windows.Point pt,string page,Color color,int fontsize) * { * if (strokeidx < 0) * return; * Init(); * XmlNode pointnode = FindTypeNode(page, type, strokeidx); * if (pointnode != null) * { * XmlElement New\_X = xmlDoc.CreateElement("Pt\_X"); * New\_X.InnerText = pt.X.ToString(); * pointnode.AppendChild(New\_X); * XmlElement New\_Y = xmlDoc.CreateElement("Pt\_Y"); * New\_Y.InnerText = pt.Y.ToString(); * pointnode.AppendChild(New\_Y); * xmlDoc.AppendChild(DrawElement); * xmlDoc.Save(DrawSave); * return; * } * XmlElement DlistElement = null; * XmlNodeList pagelist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']"); * XmlElement Page = null; * if (pagelist[0] != null) * { * Page = (XmlElement)pagelist[0]; * } * else * { * DlistElement = xmlDoc.CreateElement("DrawList"); * Page = xmlDoc.CreateElement("Page"); * } * XmlAttribute PageAttribute = xmlDoc.CreateAttribute("Value"); * Page.SetAttributeNode(PageAttribute); * PageAttribute.Value = page; * XmlElement TypeElement = xmlDoc.CreateElement("Type"); * XmlAttribute TypeAttribute = xmlDoc.CreateAttribute("Value"); * TypeElement.SetAttributeNode(TypeAttribute); * TypeAttribute.Value = type; * XmlElement IdxElement = xmlDoc.CreateElement("Index"); * XmlAttribute IdxAttribute = xmlDoc.CreateAttribute("Value"); * IdxElement.SetAttributeNode(IdxAttribute); * IdxAttribute.Value = strokeidx.ToString(); * XmlElement ColorElement = xmlDoc.CreateElement("Color"); * ColorElement.InnerText = color.ToString(); * IdxElement.AppendChild(ColorElement); * XmlElement FontSizeElement = xmlDoc.CreateElement("FontSize"); * FontSizeElement.InnerText = fontsize.ToString(); * IdxElement.AppendChild(FontSizeElement); * XmlElement PtElement = xmlDoc.CreateElement("Point"); * IdxElement.AppendChild(PtElement); * XmlElement Pt\_X = xmlDoc.CreateElement("Pt\_X"); * Pt\_X.InnerText = pt.X.ToString(); * PtElement.AppendChild(Pt\_X); * XmlElement Pt\_Y = xmlDoc.CreateElement("Pt\_Y"); * Pt\_Y.InnerText = pt.Y.ToString(); * PtElement.AppendChild(Pt\_Y); * TypeElement.AppendChild(IdxElement); * Page.AppendChild(TypeElement); * if (DlistElement != null) * { * DlistElement.AppendChild(Page); * DrawElement.AppendChild(DlistElement); * } * xmlDoc.AppendChild(DrawElement); * xmlDoc.Save(DrawSave); * } * static public void Xml\_Save(string type, string id, double left, double top, double width, double height,string page) * { * Init(); * XmlElement DlistElement = null; * XmlNodeList pagelist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']"); * XmlElement Page = null; * if (pagelist[0] != null) * { * Page = (XmlElement)pagelist[0]; * } * else * { * DlistElement = xmlDoc.CreateElement("DrawList"); * Page = xmlDoc.CreateElement("Page"); * } * XmlAttribute PageAttribute = xmlDoc.CreateAttribute("Value"); * Page.SetAttributeNode(PageAttribute); * PageAttribute.Value = page; * XmlElement TypeElement = xmlDoc.CreateElement("Type"); * XmlAttribute TypeAttribute = xmlDoc.CreateAttribute("Value"); * TypeElement.SetAttributeNode(TypeAttribute); * TypeAttribute.Value = type; * XmlElement IdElement = xmlDoc.CreateElement("Id"); * XmlAttribute IdAttribute = xmlDoc.CreateAttribute("Value"); * IdElement.SetAttributeNode(IdAttribute); * IdAttribute.Value = id; * XmlElement LeftElement = xmlDoc.CreateElement("Left"); * LeftElement.InnerText = left.ToString(); * IdElement.AppendChild(LeftElement); * XmlElement TopElement = xmlDoc.CreateElement("Top"); * TopElement.InnerText = top.ToString(); * IdElement.AppendChild(TopElement); * XmlElement WidthElement = xmlDoc.CreateElement("Width"); * WidthElement.InnerText = width.ToString(); * IdElement.AppendChild(WidthElement); * XmlElement HeightElement = xmlDoc.CreateElement("Height"); * HeightElement.InnerText = height.ToString(); * IdElement.AppendChild(HeightElement); * TypeElement.AppendChild(IdElement); * Page.AppendChild(TypeElement); * if (DlistElement != null) * { * DlistElement.AppendChild(Page); * DrawElement.AppendChild(DlistElement); * } * xmlDoc.AppendChild(DrawElement); * xmlDoc.Save(DrawSave); * } * static public void Remove(string page,int idx,int length) * { * Init(); * XmlNodeList idxlist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']/Type/Index"); * //첫번째 인덱스 * if (idx == 0) * { * XmlNode deletenode = idxlist[idx]; * foreach (XmlNode idxnode in idxlist) * { * if (idxnode == idxlist[idx]) * continue; * idxnode.Attributes["Value"].InnerText = (idx++).ToString(); * } * deletenode.ParentNode.ParentNode.RemoveChild(deletenode.ParentNode); * } * //중간 인덱스 * else if (idx > 0 && idx < length - 1) * { * XmlNode deletenode = idxlist[idx]; * foreach (XmlNode idxnode in idxlist) * { * if (idxnode.Attributes["Value"].InnerText == "0") * continue; * if (idxnode == idxlist[idx]) * continue; * idxnode.Attributes["Value"].InnerText = (idx++).ToString(); * } * deletenode.ParentNode.ParentNode.RemoveChild(deletenode.ParentNode); * } * //마지막 인덱스 * else if (idx == length - 1) * { * idxlist[idx].ParentNode.ParentNode.RemoveChild(idxlist[idx].ParentNode); * } * xmlDoc.AppendChild(DrawElement); * xmlDoc.Save(DrawSave); * } * static public void Remove(string page, string id) * { * Init(); * XmlNodeList typelist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']/Type[@Value='Rectangle']/Id[@Value='"+id+"']"); * typelist[0].ParentNode.ParentNode.RemoveChild(typelist[0].ParentNode); * xmlDoc.AppendChild(DrawElement); * xmlDoc.Save(DrawSave); * } * static public ReadData[] XML\_Read(string page) * { * Init(); * ArrayList data = new ArrayList(); * XmlNodeList idxlist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']/Type/Index"); * XmlNodeList typelist = xmlDoc.SelectNodes("//Page[@Value='" + page + "']/Type[@Value='Rectangle']/Id"); * if (idxlist[0] != null) * { * for (int i = 0; i < idxlist.Count; i++) * { * XmlNode pointnode = idxlist[i].LastChild; * ArrayList ptlist = new ArrayList(); * Point pt = new Point(); * int j = 0; * foreach (XmlNode node in pointnode.ChildNodes) * { * j++; * if (j % 2 == 0) * { * pt.Y = double.Parse(node.InnerText); * ptlist.Add(pt); * } * else * { * pt.X = double.Parse(node.InnerText); * } * } * ReadData readdata = new ReadData(); * readdata.Pts = (Point[])ptlist.ToArray(typeof(Point)); * Color color = (Color)ColorConverter.ConvertFromString(idxlist[i]["Color"].InnerText); * readdata.Color = color; * readdata.FontSize = int.Parse(idxlist[i]["FontSize"].InnerText); * readdata.Type = idxlist[i].ParentNode.Attributes["Value"].InnerText; * data.Add(readdata); * } * if (typelist[0] != null) * { * foreach (XmlNode node in typelist) * { * double left = double.Parse(node["Left"].InnerText); * double top = double.Parse(node["Top"].InnerText); * double width = double.Parse(node["Width"].InnerText); * double height = double.Parse(node["Height"].InnerText); * string id = node.Attributes["Value"].InnerText; * ReadData readdata = new ReadData(); * readdata.Text = new TextControl * { * BorderBrush = System.Windows.Media.Brushes.LightBlue, * BorderThickness = new Thickness(2), * Background = System.Windows.Media.Brushes.LightGreen, * Opacity = 0.5, * Width = width, * Height = height, * ID = id * }; * readdata.Text.SetValue(MyInkCanvas.LeftProperty, left); * readdata.Text.SetValue(MyInkCanvas.TopProperty, top); * readdata.Type = "Rectangle"; * readdata.Text.Text\_Changed += PdfWin.Text\_Changed; * data.Add(readdata); * } * } * return (ReadData[])data.ToArray(typeof(ReadData)); * } * else * { * if (typelist[0] != null) * { * foreach (XmlNode node in typelist) * { * double left = double.Parse(node["Left"].InnerText); * double top = double.Parse(node["Top"].InnerText); * double width = double.Parse(node["Width"].InnerText); * double height = double.Parse(node["Height"].InnerText); * string id = node.Attributes["Value"].InnerText; * ReadData readdata = new ReadData(); * readdata.Text = new TextControl * { * BorderBrush = System.Windows.Media.Brushes.LightBlue, * BorderThickness = new Thickness(2), * Background = System.Windows.Media.Brushes.LightGreen, * Opacity = 0.5, * Width = width, * Height = height, * ID = id * }; * readdata.Text.SetValue(MyInkCanvas.LeftProperty, left); * readdata.Text.SetValue(MyInkCanvas.TopProperty, top); * readdata.Type = "Rectangle"; * readdata.Text.Text\_Changed += PdfWin.Text\_Changed; * data.Add(readdata); * } * } * return (ReadData[])data.ToArray(typeof(ReadData)); * } * } * } * public class ReadData * { * public Color Color * { * get; * set; * } * public int FontSize * { * get; * set; * } * public Point[] Pts * { * get; * set; * } * public TextControl Text * { * get; * set; * } * public string Type * { * get; * set; * } * } |

* 서비스 코드
* 콜백 함수 내에서 서비스 클라이언트 사용하기

|  |
| --- |
| * [CallbackBehavior(ConcurrencyMode = ConcurrencyMode.Multiple)] * public class Service : ISIUCallback, IDisposable |
| //콜백함수  public void SendResponse(string sender, string file)  {  if (System.Windows.MessageBox.Show(sender + "님이 파일 전송을 요청합니다.", "File Transfer", MessageBoxButton.YesNo)  == MessageBoxResult.Yes)  {  System.Windows.Forms.SaveFileDialog dlg = new System.Windows.Forms.SaveFileDialog()  {  Title = "Select a file to download",  RestoreDirectory = true,  FileName = file  };  if (dlg.ShowDialog() == System.Windows.Forms.DialogResult.OK)  {  this.filepath = dlg.FileName;  this.file = file;  }  }  else  this.client.DeleteFile(file, -1);   * } |

**3.1.2 서비스**

- 기본적으로 usb 인식을 통한 프로세스 실행을 서비스로 구동

- 서비스에서 프로세스 구동은 백그라운드에서 작동

- 세션을 구분하여 사용자의 눈에 보여줌

|  |
| --- |
| private static uint STANDARD\_RIGHTS\_REQUIRED = 0x000F0000;  private static uint STANDARD\_RIGHTS\_READ = 0x00020000;  private static uint TOKEN\_ASSIGN\_PRIMARY = 0x0001;  private static uint TOKEN\_DUPLICATE = 0x0002;  private static uint TOKEN\_IMPERSONATE = 0x0004;  private static uint TOKEN\_QUERY = 0x0008;  private static uint TOKEN\_QUERY\_SOURCE = 0x0010;  private static uint TOKEN\_ADJUST\_PRIVILEGES = 0x0020;  private static uint TOKEN\_ADJUST\_GROUPS = 0x0040;  private static uint TOKEN\_ADJUST\_DEFAULT = 0x0080;  private static uint TOKEN\_ADJUST\_SESSIONID = 0x0100;  private static uint TOKEN\_READ = (STANDARD\_RIGHTS\_READ | TOKEN\_QUERY);  private static uint TOKEN\_ALL\_ACCESS = (STANDARD\_RIGHTS\_REQUIRED | TOKEN\_ASSIGN\_PRIMARY | TOKEN\_DUPLICATE | TOKEN\_IMPERSONATE | TOKEN\_QUERY | TOKEN\_QUERY\_SOURCE | TOKEN\_ADJUST\_PRIVILEGES | TOKEN\_ADJUST\_GROUPS | TOKEN\_ADJUST\_DEFAULT | TOKEN\_ADJUST\_SESSIONID);  private const uint NORMAL\_PRIORITY\_CLASS = 0x0020;  private const uint CREATE\_UNICODE\_ENVIRONMENT = 0x00000400;  private const uint MAX\_PATH = 260;  private const uint CREATE\_NO\_WINDOW = 0x08000000;  private const uint INFINITE = 0xFFFFFFFF;  [StructLayout(LayoutKind.Sequential)]  public struct SECURITY\_ATTRIBUTES  {  public int nLength;  public IntPtr lpSecurityDescriptor;  public int bInheritHandle;  }  public enum SECURITY\_IMPERSONATION\_LEVEL  {  SecurityAnonymous,  SecurityIdentification,  SecurityImpersonation,  SecurityDelegation  }  public enum TOKEN\_TYPE  {  TokenPrimary = 1,  TokenImpersonation  }  public enum WTS\_CONNECTSTATE\_CLASS  {  WTSActive,  WTSConnected,  WTSConnectQuery,  WTSShadow,  WTSDisconnected,  WTSIdle,  WTSListen,  WTSReset,  WTSDown,  WTSInit  }  [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]  public struct STARTUPINFO  {  public Int32 cb;  public string lpReserved;  public string lpDesktop;  public string lpTitle;  public Int32 dwX;  public Int32 dwY;  public Int32 dwXSize;  public Int32 dwYSize;  public Int32 dwXCountChars;  public Int32 dwYCountChars;  public Int32 dwFillAttribute;  public Int32 dwFlags;  public Int16 wShowWindow;  public Int16 cbReserved2;  public IntPtr lpReserved2;  public IntPtr hStdInput;  public IntPtr hStdOutput;  public IntPtr hStdError;  }  [StructLayout(LayoutKind.Sequential)]  internal struct PROCESS\_INFORMATION  {  public IntPtr hProcess;  public IntPtr hThread;  public int dwProcessId;  public int dwThreadId;  }  [StructLayout(LayoutKind.Sequential)]  private struct WTS\_SESSION\_INFO  {  public Int32 SessionID;  [MarshalAs(UnmanagedType.LPStr)]  public String pWinStationName;  public WTS\_CONNECTSTATE\_CLASS State;  }  [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern uint WTSGetActiveConsoleSessionId();  [DllImport("wtsapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern bool WTSQueryUserToken(int sessionId, out IntPtr tokenHandle);  [DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]  public extern static bool DuplicateTokenEx(IntPtr existingToken, uint desiredAccess, IntPtr tokenAttributes, SECURITY\_IMPERSONATION\_LEVEL impersonationLevel, TOKEN\_TYPE tokenType, out IntPtr newToken);  [DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern bool CreateProcessAsUser(IntPtr token, string applicationName, string commandLine, ref SECURITY\_ATTRIBUTES processAttributes, ref SECURITY\_ATTRIBUTES threadAttributes, bool inheritHandles, uint creationFlags, IntPtr environment, string currentDirectory, ref STARTUPINFO startupInfo, out PROCESS\_INFORMATION processInformation);  [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern bool CloseHandle(IntPtr handle);  [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern int GetLastError();  [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern int WaitForSingleObject(IntPtr token, uint timeInterval);  [DllImport("wtsapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern int WTSEnumerateSessions(System.IntPtr hServer, int Reserved, int Version, ref System.IntPtr ppSessionInfo, ref int pCount);  [DllImport("userenv.dll", CharSet = CharSet.Auto, SetLastError = true)]  static extern bool CreateEnvironmentBlock(out IntPtr lpEnvironment, IntPtr hToken, bool bInherit);  [DllImport("wtsapi32.dll", ExactSpelling = true, SetLastError = false)]  public static extern void WTSFreeMemory(IntPtr memory);  [DllImport("userenv.dll", SetLastError = true)]  [return: MarshalAs(UnmanagedType.Bool)]  static extern bool DestroyEnvironmentBlock(IntPtr lpEnvironment);  private PROCESS\_INFORMATION processInfo\_;  public static IntPtr GetCurrentUserToken()  {  IntPtr currentToken = IntPtr.Zero;  IntPtr primaryToken = IntPtr.Zero;  IntPtr WTS\_CURRENT\_SERVER\_HANDLE = IntPtr.Zero;  int dwSessionId = 0;  IntPtr hUserToken = IntPtr.Zero;  IntPtr hTokenDup = IntPtr.Zero;  IntPtr pSessionInfo = IntPtr.Zero;  int dwCount = 0;  WTSEnumerateSessions(WTS\_CURRENT\_SERVER\_HANDLE, 0, 1, ref pSessionInfo, ref dwCount);  Int32 dataSize = Marshal.SizeOf(typeof(WTS\_SESSION\_INFO));  Int32 current = (int)pSessionInfo;  for (int i = 0; i < dwCount; i++)  {  WTS\_SESSION\_INFO si = (WTS\_SESSION\_INFO)Marshal.PtrToStructure((System.IntPtr)current, typeof(WTS\_SESSION\_INFO));  if (WTS\_CONNECTSTATE\_CLASS.WTSActive == si.State)  {  dwSessionId = si.SessionID;  break;  }  current += dataSize;  }  WTSFreeMemory(pSessionInfo);  bool bRet = WTSQueryUserToken(dwSessionId, out currentToken);  if (bRet == false)  {  return IntPtr.Zero;  }  bRet = DuplicateTokenEx(currentToken, TOKEN\_ASSIGN\_PRIMARY | TOKEN\_ALL\_ACCESS, IntPtr.Zero, SECURITY\_IMPERSONATION\_LEVEL.SecurityImpersonation, TOKEN\_TYPE.TokenPrimary, out primaryToken);  if (bRet == false)  {  return IntPtr.Zero;  }  return primaryToken;  }  public void Run(string serial)  {  IntPtr primaryToken = GetCurrentUserToken();  if (primaryToken == IntPtr.Zero)  {  return;  }  STARTUPINFO StartupInfo = new STARTUPINFO();  processInfo\_ = new PROCESS\_INFORMATION();  StartupInfo.cb = Marshal.SizeOf(StartupInfo);  SECURITY\_ATTRIBUTES Security1 = new SECURITY\_ATTRIBUTES();  SECURITY\_ATTRIBUTES Security2 = new SECURITY\_ATTRIBUTES();  //string command = "\"" + processPath\_ + "\"";  //string command = Application.StartupPath + "[\\EDMLogin.exe](file://\\EDMLogin.exe\)";  string command = @"C:\Users\77\Desktop\시연\최종합체 0904 - 1차 수정\UserWallPaper\WpfApplication1\bin\Debug\UserWallPaper.exe";  command += " " + serial;  //string command = "C:\\EDMLoginTest\\EDMLogin.exe";  //if ((arguments\_ != null) && (arguments\_.Length != 0))  //{  // command += " " + arguments\_;  //}  IntPtr lpEnvironment = IntPtr.Zero;  bool resultEnv = CreateEnvironmentBlock(out lpEnvironment, primaryToken, false);  if (resultEnv != true)  {  int nError = GetLastError();  }  CreateProcessAsUser(primaryToken, null, command, ref Security1, ref Security2, false, CREATE\_NO\_WINDOW | NORMAL\_PRIORITY\_CLASS | CREATE\_UNICODE\_ENVIRONMENT, lpEnvironment, null, ref StartupInfo, out processInfo\_);  DestroyEnvironmentBlock(lpEnvironment);  CloseHandle(primaryToken);  } |

**5절 결론 및 개선 사항**

* **기대 효과**

- 공용 pc를 학생 자신만의 개인 pc처럼 구성이 가능하여 학생 스스로의 학습

분위기를 향상시킴

- usb 하나만으로 수업 준비와 강의자료를 열람할 수 있어 개인의 학습효과 증가

- 추가적인 장비 비용과 라이선스 비용이 감소되어 학교 측의 부담을 줄여줌

* **개선방안**

- 좀 더 다양한 문서를 출력할 수 있는 뷰어

- 강사의 화면 노고하 및 전송

- 바탕화면 및 뷰어 UI Design 개선

- 바탕화면 기능 개선 및 추가

**6절 참고 문헌**

**사이트**

<http://msdn.microsoft.com/en-us/>

[http://www.codeproject.com](http://www.codeproject.com/)

<http://www.devpia.com/>

<http://www.hoons.net/>

**사용언어, 사용소프트웨어**

운영체제 : Windows 7 Ultimate, .NET FrameWork 4.5

소프트웨어 : Visual Studio 2012, WPF, WCF

언어 : C#