University of Macau

Faculty of Science and Technology



COIS704 Current Development in Database

Course Project: Online Air Ticket Query Web Application

**Supervisor:** Prof. Zhiguo GONG

**Academic Year:** 2015-2016

**Major:** Computer Science

**Student:** M-B4-5441, FENG QIYING

[M-B4-5442, LV RAN](https://ummoodle.umac.mo/user/view.php?id=23157&course=2329)

M-B4-5444, WANG WEI

Content

[1. System Description 2](#_Toc436567341)

[2. System Architecture 2](#_Toc436567342)

[3. Implementation 4](#_Toc436567343)

[3.1 User Interface 4](#_Toc436567344)

[3.2 Functions and Coding 5](#_Toc436567345)

[4. Conclusion and Problems 10](#_Toc436567346)

[5. References 10](#_Toc436567348)

# System Description

There are many websites provide the air ticket information include the basic information of flight and the price. Our system will integrate 5 data source to one global database. The 5 data sources are Qunaer, CTRIP, [China](app:ds:China) [Eastern](app:ds:Eastern) [Airlines](app:ds:Airlines), [China](app:ds:China) Southern [Airlines](app:ds:Airlines), and web services. Our system provide an interface for query the collected air ticket information base on the input filter like depart city, arrive city and the date. Because Internet and web browsers are widely used all over the world, so we design the system as Web-Based. We presents the implementation of the system by visual studio 2013 C# with .net 4.5. And the database is SQL Server express.

# System Architecture

If design a total new system, we prefer to use the top to down. But, for this project, we should collect data from different data source, we will use the other method named Multi-Databases. For the 5 data sources, we found that there are 4 types data structure: xml, Json, Soap and Pure HTML. So we will use the local schema for each source, then convert to global schema to integrate the different sources for query function.

Then, we define the global schema as class Flight:

|  |  |
| --- | --- |
| Field Name | Description |
| From | 出发城市 |
| To | 到达城市 |
| Departure | 出发日期 |
| DateSource | 数据来源网站 |
| Airline | 航空公司 |
| FlightNo | 航班编号 |
| DeptTime | 起飞时间 |
| ArriveTime | 到达时间 |
| TotalTime | 航程时长 |
| FirstClass | 头等舱 |
| Business | 公务舱 |
| Economy | 经济舱 |
| Price | 价格 |
| Remark | 其他信息 |

For the global schema, we will create a global class to retrieve detail records and display to the webpage.

The following figure is the architecture of this system.



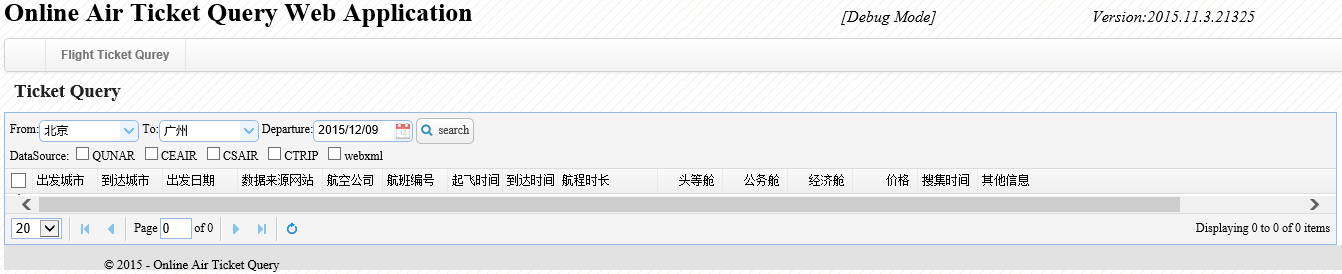
# Implementation

Our particular concern after system architecture is implementation.

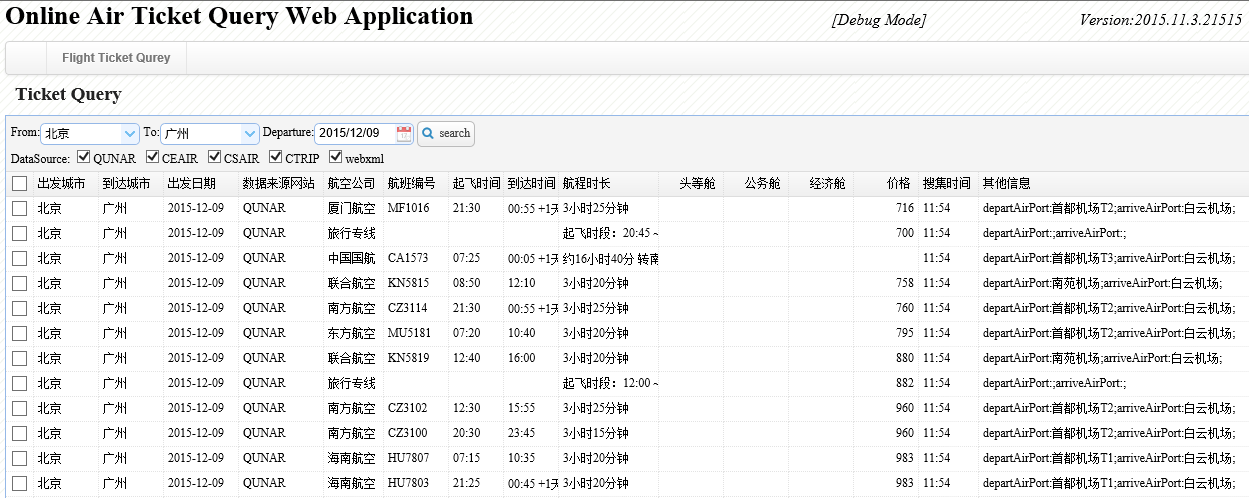
After study the query capabilities and provided price information method of each data source, we found that the Qunaer and CE airline are use Ajax technology. For these two web site, the traditional crawl methods cannot finish the job. In asp.net, we can use a controller named WebBrowser to simulate the Ajax call, then we can get the source html to parse and get what we wanted.

## 3.1 User Interface

A screen shot of the system is depicted in following Figure.



The following figure show that result after integrate data.



At this page, each columns can sorted by click the columns title. For example, if we want to all flights by depart time.



## 3.2 Functions and Coding

1. For the source contain Ajax, we use the following class named PageSnatch to get the page html. The class use a WebBrowser to simulate the Ajax.

|  |
| --- |
| using (WebBrowser browser = new WebBrowser())  {  browser.ScriptErrorsSuppressed = false;  DateTime startTime = DateTime.Now;  bool isbusy = true;  int length = 0;  browser.Navigate(url);  while (browser.ReadyState != WebBrowserReadyState.Complete)  {  Application.DoEvents();  System.Threading.Thread.Sleep(interval);  double t = Math.Ceiling((DateTime.Now - startTime).TotalSeconds);  if (t >= timeout)  {  throw new Exception("Visiting about new exception delay, since the setting is timeout");  }  }  while (hitCount < 4)  {  double t = Math.Ceiling((DateTime.Now - startTime).TotalSeconds);  if (t >= timeout)  {  throw new Exception("Visiting about new exception delay, since the setting is timeout");  }  BrowserEventHandler browserEventHanler = delegate() { isbusy = !browser.IsBusy; };  browser.Invoke(browserEventHanler);  if (browser.Document.All[flightHtmlElementID] != null)  {  int len = 0;  if (!string.IsNullOrEmpty(browser.Document.All[flightHtmlElementID].InnerHtml))  len = browser.Document.All[flightHtmlElementID].InnerHtml.Length;  if (len == length)  {  hitCount++;  }  else  {  hitCount = 0; length = len;  }  }  if (!string.IsNullOrEmpty(browser.Document.All[flightHtmlElementID].InnerHtml))  length = browser.Document.All[flightHtmlElementID].InnerHtml.Length;  System.Threading.Thread.Sleep(interval);  }  if (browser.Document.All[flightHtmlElementID] != null)  {  gethtml = browser.Document.All[flightHtmlElementID].InnerHtml;  }  } |

After get the html source, we try to use the Regex Expression to parse the HTML at first, then we found that is quite impossible because the page layouts are very different. Fortunately, we get a third party and open source library HtmlAgilityPack, this library can parse the page source to HTML node. For example, we can get the inner text of HTML node use the following function:

|  |
| --- |
| private string GetInnerText(HtmlNode checkNode)  {  string strInnerText = string.Empty;  if (checkNode != null)  strInnerText = checkNode.InnerText.Trim(new char[] { ' ', '\r', '\n' });  return strInnerText;  } |

During the debug, we found that the html node may not exists, so we must check the node exist before get the inner text or sub html node.

|  |
| --- |
| private HtmlNode GetHtmlNode(HtmlNode checkNode, string nodeXpath, int nodeIndex)  {  HtmlNode nodeResult = null;  if (checkNode != null)  {  HtmlNodeCollection nodeList = checkNode.SelectNodes(nodeXpath);  if (nodeList.Count > nodeIndex)  {  nodeResult = nodeList[nodeIndex];  }  }  return nodeResult;  } |

The detail technology and full source please refer to functions with name “QUNAR\_Get” and “CEAIR\_Get”.

1. For the source type is xml, this is much easy to parse. We just need study the xml structure once, then we can get the detail information by xml node. Use China Southern Airlines as example, the following code show the process.

|  |
| --- |
| List<Flight> lstFlight = new List<Flight>();  DateTime dtDepart = DateTime.Parse(departDate);  string strUrl = string.Format("http://b2c.csair.com/B2C40/detail-{0}{1}-{2}-1-0-0-0-1-0-0-0-1-0.g2c",  fromCity.C\_CODE, toCity.C\_CODE, dtDepart.ToString("yyyyMMdd"));  XmlDocument doc = new XmlDocument();  doc.Load(strUrl);  XmlHelper xmlHelper = new XmlHelper(doc);  XmlNodeList nodelist = xmlHelper.GetXmlNodeListByXpath("FLIGHTS/SEGMENT/DATEFLIGHT/DIRECTFLIGHT/FLIGHT");  foreach (XmlNode node in nodelist)  {  Flight f = new Flight();  f.C\_DateSource = "CS AIR";  f.C\_From = fromCity.C\_NAME;  f.C\_To = toCity.C\_NAME;  f.C\_Departure = departDate;  f.C\_FlightNo = XmlNodeHelper.ParseByNode(node, "FLIGHTNO");  f.C\_Airline = XmlNodeHelper.ParseByNode(node, "AIRLINE");  f.C\_DEPTIME = XmlNodeHelper.ParseByNode(node, "DEPTIME");  f.C\_ARRTIME = XmlNodeHelper.ParseByNode(node, "ARRTIME");  f.C\_TotalTime = XmlNodeHelper.ParseByNode(node, "TIMEDURINGFLIGHT\_en");  StringBuilder sbPriceInfo = new StringBuilder();  XmlNodeList xnlPrice = node.SelectNodes("CABINS/CABIN");  foreach (XmlNode childNodePrice in xnlPrice)  {  string nodeName = XmlNodeHelper.ParseByNode(childNodePrice, "NAME");  string strPrice = XmlNodeHelper.ParseByNode(childNodePrice, "ADULTPRICE");  if (nodeName.Equals("P") && !string.IsNullOrEmpty(strPrice))  {  f.C\_FirstClass = Convert.ToDecimal(strPrice);  }  else if (nodeName.Equals("Y") && !string.IsNullOrEmpty(strPrice))  {  f.C\_Economy = Convert.ToDecimal(strPrice);  }  else if (nodeName.Equals("D") && !string.IsNullOrEmpty(strPrice))  {  f.C\_Business = Convert.ToDecimal(strPrice);  }  else  {  sbPriceInfo.AppendFormat("nodeName:{0}->ADULTPRICE:{1}->DISCOUNT:{2}->ADULTFAREBASIS:{3}->GBADULTPRICE:{4}"  + "->BRANDTYPE:{5}->MILEAGESTANDARD:{6}",  nodeName, XmlNodeHelper.ParseByNode(childNodePrice, "ADULTPRICE") ?? string.Empty  , XmlNodeHelper.ParseByNode(childNodePrice, "DISCOUNT") ?? string.Empty  , XmlNodeHelper.ParseByNode(childNodePrice, "ADULTFAREBASIS") ?? string.Empty  , XmlNodeHelper.ParseByNode(childNodePrice, "GBADULTPRICE") ?? string.Empty  , XmlNodeHelper.ParseByNode(childNodePrice, "BRANDTYPE") ?? string.Empty  , XmlNodeHelper.ParseByNode(childNodePrice, "MILEAGESTANDARD") ?? string.Empty);  }  }  f.C\_Remark = sbPriceInfo.ToString();  lstFlight.Add(f);  }  return lstFlight; |

1. For the source type is Json, we use another third party library Newtonsoft.Json, after we define the class base on the Json structure, this library can easy to Deserialize the Json string to class.

|  |
| --- |
| var objReturnResult = JsonConvert.DeserializeObject<ReturnResult>(jsonString);  if (objReturnResult.IsSucceed)  {  if (objReturnResult.FlightRoutes.Count > 0)  {  FlightRoute flightRoute = objReturnResult.FlightRoutes[0];  foreach (var flightInfo in flightRoute.FlightsList)  {  Flight f = new Flight();  f.C\_DateSource = "CTRIP API";  f.C\_From = flightRoute.DCityName;  f.C\_To = flightRoute.ACityName;  f.C\_Departure = flightRoute.DDate;  f.C\_FlightNo = flightInfo.Flight;  f.C\_Airline = flightInfo.AirlineCode;  f.C\_DEPTIME = flightInfo.TakeOffTime.Substring(11, 5);  f.C\_ARRTIME = flightInfo.ArriveTime.Substring(11, 5);  FlightClass firstFlightClass = flightInfo.FlightClassList.Find(ff => ff.Class.Equals("F", StringComparison.CurrentCultureIgnoreCase));  if (firstFlightClass != null)  f.C\_FirstClass = Convert.ToDecimal(firstFlightClass.Price);  FlightClass economyFlightClass = flightInfo.FlightClassList.Find(ff => ff.Class.Equals("Y", StringComparison.CurrentCultureIgnoreCase));  if (economyFlightClass != null)  f.C\_Economy = Convert.ToDecimal(economyFlightClass.Price);  lstFlight.Add(f);  }  }  } |

1. For the source type is SOAP, after add web reference of the web service url, the return define of this type will clearly showed.

|  |
| --- |
| DateTime dtDepart = DateTime.Parse(departDate);  AirTicketQuery.DomesticAirline.DomesticAirline wsAirLine = new DomesticAirline.DomesticAirline();  DataSet dsFlight = wsAirLine.getDomesticAirlinesTime(fromCity.C\_NAME, toCity.C\_NAME, dtDepart.ToString("yyyy-MM-dd"), string.Empty);  foreach (DataRow dr in dsFlight.Tables[0].Rows)  {  Flight f = new Flight();  f.C\_DateSource = "webxml";  f.C\_From = fromCity.C\_NAME;  f.C\_To = toCity.C\_NAME;  f.C\_Departure = departDate;  f.C\_Airline = dr["Company"].ToString();  f.C\_FlightNo = dr["AirlineCode"].ToString();  f.C\_DEPTIME = dr["StartTime"].ToString();  f.C\_ARRTIME = dr["ArriveTime"].ToString();  f.C\_Remark = string.Format("出发机场:{0}->到达机场:{1}->机型:{2}->经停:{3}->飞行周期（星期）:{4}",  dr["StartDrome"], dr["ArriveDrome"], dr["Mode"], dr["AirlineStop"], dr["Week"]);  lstFlight.Add(f);  } |

1. At last, we provide a main function for query interface. According to the query filter, we will check the collected time of source whether less than 15mins. If yes, we just get the records from database, else we will retrieve the data again.

For enhance the retrieve efficiency, we use multiple thread to parallel collect data.

|  |
| --- |
| Task<List<Flight>>[] tasksList = new Task<List<Flight>>[noOfTask];  int taskIndex = 0;  if (needGetCSAIR)  tasksList[taskIndex++] = Task<List<Flight>>.Factory.StartNew(() => this.CSAIR\_Get(fromCity, toCity, strDeparture));  if (needGetCTRIP)  tasksList[taskIndex++] = Task<List<Flight>>.Factory.StartNew(() => this.CTRIP\_Get(fromCity, toCity, strDeparture));  if (needGetWS)  tasksList[taskIndex++] = Task<List<Flight>>.Factory.StartNew(() => this.WS\_Get(fromCity, toCity, strDeparture));  if (noOfTask > 0)  Task.WaitAll(tasksList, 50 \* 1000);  List<Flight> lstFlight = new List<Flight>();  foreach (var item in tasksList)  {  lstFlight.AddRange(item.Result);  }  if (lstFlight.Count > 0)  dbi.WriteData(lstFlight.ToDataTable(), "FlightInfo"); |

# Conclusion and Problems



In this project, we integrated different data source like xml, webservics, json and html source. The system is only finished in the first phrase. It has lots of parts that need to be improved. We list out some main points need to be solved in the future:

1. An interface to maintain the city list should provide.
2. Need study the price description of each source and the price of ticket can split to part by the flight class.
3. When get the page source which used Ajax, the timeout exception will occurred sometime, we should find a stable solution for this problem.
4. Check how to combine the thread that used to get the Ajax source to the thread list. [In](http://cn.bing.com/dict/search?q=In&FORM=BDVSP6&mkt=zh-cn) [current](http://cn.bing.com/dict/search?q=current&FORM=BDVSP6&mkt=zh-cn) [implementation](http://cn.bing.com/dict/search?q=implementation&FORM=BDVSP6&mkt=zh-cn), we must waiting the thread finish before get the other source data.

# References

[1] M. Tamer Ozsu: *Principles of Distribute Database Systems*, Third Edition, Springer.

[2] 范轩苗, 郑宁, 范渊： 一种基于Ajax的爬虫模型的设计与实现，Computer Applications and Software，V01．27 No．1, Jan．2010.

[3] 钱程, 阳小兰: 一种支持Ajax框架的网络爬虫的设计与实现, Computer & Digital Engineering,V01．40 No．4, 2012.

[4] 张成奇．支持Ajax 的Deep Web 爬虫设计与实现[硕士学位论文]．软件学院，上海交通大学，2009．