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-B4-5444, WANG WEI

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

M-B4-5441, FENG QIYING

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

Air fight is an important transport mode. People needs to find the best price and the appropriate time of the air ticket for which we want to implement a system providing basic information of air fight of different air company as group project. Our system integrates 5 data sources into 1 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 provides an interface for querying the collected air ticket information basing on the input filter like depart city, arrive city and the date. We designed the system into a website for the reason that internet and web browsers are convenience and widely used all over the world. We implemented the fight information query system with the environment visual studio 2013 C# and .net 4.5 and the SQL Server express.

# System Architecture

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

Then, we defined the global schema as class Flight given by table 1:

|  |  |
| --- | --- |
| 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 |

Table 1. The global schema as class Flight

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

The following figure 1 is the architecture of this system.

Figure 1. Air fight information query system

# Implementation

Our particular concerns after system architecture is implementation.

After studying 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 2.

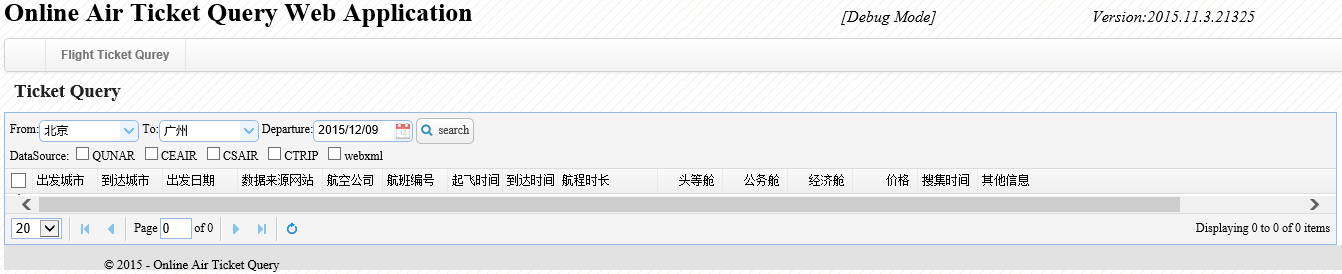


Figure 2 online air ticket query web application

The following figure 3 show that result after integrate data.

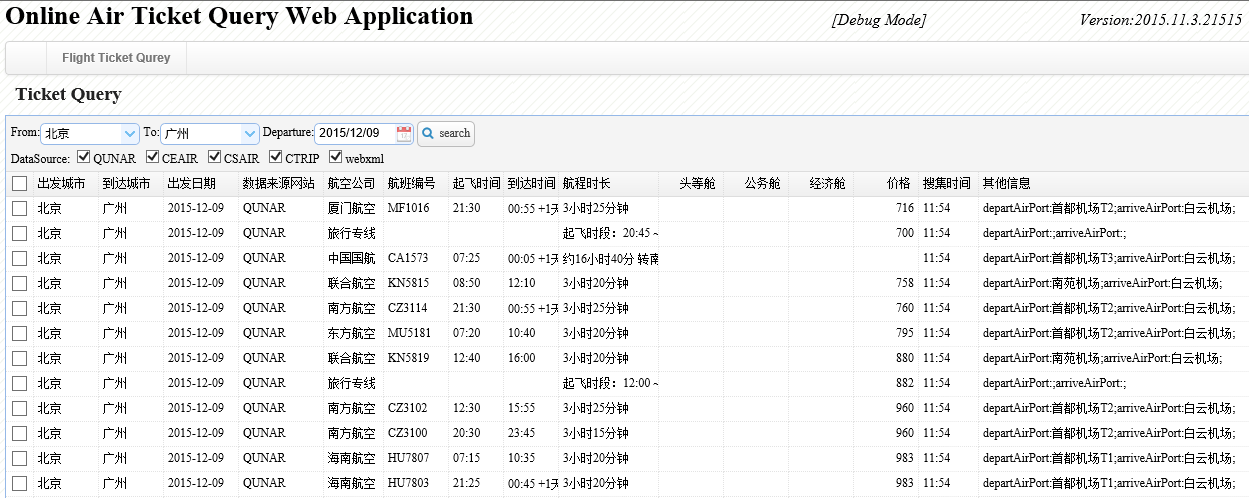


Figure 3. The result after integrating data

At this page, all columns can be sorted by clicking the columns title are given by the figure 4. For example, if we want to query all flights by departing time. Then the response of this system is given by figure 4.



Figure 4. Example

## 3.2 Functions and Coding

1. For the source contains Ajax, we used the following class named PageSnatch to get the page html. The class uses 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 geting the html source, we try to use the Regex Expression to parse the HTML at the beginning, then we met a problem, that is, quite impossible because the page layouts are completely 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;  } |

For the detail of technology and full source, please refer to functions with named “QUNAR\_Get” and “CEAIR\_Get”.

1. For the xml type source, it is relatively easy to be parsed. We just need to study the xml structure once, then we can get the detail information by xml node. Using China Southern Airlines as an example, the following code shows 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 defining the class basing on the Json structure, this library can easy 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 adding web reference of the web service URL, the return defined of this type is 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 provided 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 enhancing the retrieve efficiency, we used 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, web servics, json and html source. The system is only finished in the first phrase. It has lots of places that need to be improved. We list some main points that need to be solved in the following:

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．