```
//+------------------------------------------------------------------+
//|                                C_Calendar_ForexFactoryImport.mqh |
//|                                        Copyright 2017, M Wilson. |
//|                                      https://www.algotrader.blog |
//+------------------------------------------------------------------+
#include <C_Calendar.mqh>

#property copyright "Copyright 2017, M Wilson"
#property link      "https://www.algotrader.blog"
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
class C_Calendar_ForexFactoryImport : public C_Calendar
  {
public:
   //Constructor/Destructor
   C_Calendar_ForexFactoryImport();
   C_Calendar_ForexFactoryImport(string &strImportCountryList[], const bool
boolLowImpact=False, const bool boolMediumImpact=False, const bool boolHighImpact=True,
const int intCalendarTimeOffsetSecs=0);
  ~C_Calendar_ForexFactoryImport();
   //Functions
   bool SyncCalendar(const string strCalendarData="ffcal_week_this.xml");
   datetime GetLastModifyDateFromFile(const string strCalendarData="ffcal_week_this.xml"
);
   datetime GetLastSyncDate(){return m_dtLastSync;};
protected:
   datetime m_dtLastSync;
   string m_strImportCountryList[];
   bool m_boolImportLowImpact;
   bool m_boolImportMediumImpact;
   bool m_boolImportHighImpact;
   int m_intCalendarTimeOffsetSecs;
private:
   //Functions
   bool AddMiddleXMLToArrays(string strMiddleInput);
   bool StringContainsNumbersOnly(const string strInput);
   bool CountryIsValidForImport(const string strCountry);
   bool ImpactIsValidForImport(const NewsImpact eImpact);
  };
//+------------------------------------------------------------------+
//| Constructors                                                     |
//+------------------------------------------------------------------+
C_Calendar_ForexFactoryImport::C_Calendar_ForexFactoryImport()
  {
  }
C_Calendar_ForexFactoryImport::C_Calendar_ForexFactoryImport(string &
strImportCountryList[], const bool boolLowImpact=False, const bool boolMediumImpact=
False, const bool boolHighImpact=True,const int intCalendarTimeOffsetSecs=0)
  {
  if(ArraySize(this.m_strImportCountryList)>0)  ArrayFree(this.m_strImportCountryList);
  ArrayCopy(this.m_strImportCountryList,strImportCountryList,0,0);
  this.m_boolImportLowImpact=boolLowImpact;
  this.m_boolImportMediumImpact=boolMediumImpact;
  this.m_boolImportHighImpact=boolHighImpact;
  this.m_intCalendarTimeOffsetSecs=intCalendarTimeOffsetSecs;
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
C_Calendar_ForexFactoryImport::~C_Calendar_ForexFactoryImport()
  {
  if(ArraySize(this.m_strImportCountryList)>0)  ArrayFree(this.m_strImportCountryList);
  }
//+------------------------------------------------------------------+
//+------------------------------------------------------------------+
//| Public Functions                                                 |
//+------------------------------------------------------------------+
```

```
bool C_Calendar_ForexFactoryImport::SyncCalendar(const string strCalendarData=
"ffcal_week_this.xml")
{

//This function attempts to sync this calendar with the data in the file strCalendarData.

   //<weeklyevents>
   // <event>
   //    <title>Non-Manufacturing PMI</title>
   //    <country>CNY</country>
   //    <date><![CDATA[ 12-31-2017 ]]></date>
   //    <time><![CDATA[ 12:42am ]]></time>
   // <impact><![CDATA[ Medium ]]></impact>
   // <forecast/>
   //    <previous><![CDATA[ 54.8 ]]></previous>
   // </event>
   // ...
   //</weeklyevents>

   //Source of file:   "http://www.forexfactory.com/ffcal_week_this.xml"


//WARNING: Please note that I have noticed that the xml file does not display All Day events

//All Day events are also quite often associated with events that impact ALL currencies, suc

//If I want to import something with the currency "ALL", i need to include the appropriate i

//I can import all day events from say a historic calendar.   They are generally not include

//I could add it and it would allow me to backtest switching off the calendar on these all c

   //Reset the last error
   ResetLastError();

   //Set the directory
   if(!FileIsExist(strCalendarData)) return False;

   //Open the file - must be read
   int intFileHandle=FileOpen(strCalendarData,FILE_READ|FILE_TXT);

   //Scan through the file reading the data.
   string strCarryOver="";
   if(intFileHandle!=INVALID_HANDLE)
   {
      while(FileTell(intFileHandle)<FileSize(intFileHandle))
      {
         //Read data when we have failed to find <EVENT> or </EVENT>
         string strRow = strCarryOver+FileReadString(intFileHandle);
         StringToUpper(strRow);

         //Scan through the row
         while(StringLen(strRow)>0)
         {
            //Initiate strCarryOver, so that we take nothing onto the next row.
            strCarryOver = "";

            //Attempt to find the first <Event>
            int intStart = StringFind(strRow,"<EVENT>",0);


//If we find <Event>,then remove the data from the row, up to <Event>.   Then try to find </
            if(intStart>0)
            {
               strRow=StringSubstr(strRow,intStart,StringLen(strRow)-intStart);
               intStart=0;
            }


//When we get to here, we should either have a string that does not contain <Event> and intS
            //intStart=0 and a string beginning with <Event>.
```

```
            if(intStart==0)   //We have a string beginning with <Event>
            {
               int intEnd = StringFind(strRow,"</EVENT>",0);


//If we find </Event>, then we pull out the middle bit and process the data, otherwised we c
               if(intEnd>0)
               {
                  //Extract the middle bit of <Event>....</Event>
                  string strMiddle = StringSubstr(strRow,7,intEnd-7);

                  //Now remove the bit up to </Event> from the row.
                  int intNoToRemove=StringLen(strRow)-8;
                  if(intNoToRemove>0)
                  {
                     strRow=StringSubstr(strRow,intEnd+8,intNoToRemove);
                  }
                  else
                  {
                     strRow="";
                  }

                  //Add middle bit to arrays
                  AddMiddleXMLToArrays(strMiddle);
               }
               else
//We have a string like <Event>.... , but it does not contain </Event>
               {
                  strCarryOver = strRow;
                  strRow="";
               }
            }
            else      //String does nt contain <Event>, so move onto the next row.
            {  //intStart should be <0, so we move onto the next row.
               strRow="";
               strCarryOver="";
            }
         }
      }

      //Close the file
      FileClose(intFileHandle);

      this.m_dtLastSync=TimeLocal();

   }
   else
   {
      Print(__FUNCTION__,"Failed to open file ",strCalendarData," ",GetLastError());
      return False;
   }

   return True;
}
datetime C_Calendar_ForexFactoryImport::GetLastModifyDateFromFile(const string
strCalendarData="ffcal_week_this.xml")
{

//Function returns 0 by default, unless we can successfully find the file and evaluate its l
   datetime dtReturn=0;

   //If the file doesn't exist, then return 0
   string strPath = strCalendarData;
   if(!FileIsExist(strPath))  return dtReturn;

   //Open the file
   int intFileHandle=FileOpen(strPath,FILE_READ|FILE_TXT);

   //Get the last modify date of the file.
```

```
   dtReturn = (datetime)FileGetInteger(intFileHandle,FILE_MODIFY_DATE);

   //Close the file
   FileClose(intFileHandle);

   return dtReturn;
}
//+------------------------------------------------------------------+
//| Private Functions                                                |
//+------------------------------------------------------------------+
bool C_Calendar_ForexFactoryImport::AddMiddleXMLToArrays(string strMiddleInput)
{
   //strMiddle should take a format similar to:

//<title>...</title><country>EUR</country><date><![CDATA [ 12-04-2016 ]]></date><time><![CDA

   int intStart,intEnd;
   string strTitle, strCountry,strDate,strTime, strImpact, strYear, strMonth, strDay,
strHour, strMin, strAmPm;
   NewsImpact eImpact=HighImpact;

   //Convert middle to upper case
   string strMiddle=strMiddleInput;
   StringToUpper(strMiddle);

   //********
   bool boolPrint=False;
   string strZ = strMiddle;
   StringToUpper(strZ);
   if(StringFind(strZ,"ALL",0)>0 && StringFind(strZ,"DAY",0)>0)   boolPrint=True;
   //********

   //First extract the title.
   intStart=StringFind(strMiddle,"<TITLE>",0);
   intEnd = StringFind(strMiddle,"</TITLE>",intStart);
   if(intStart>=0 && intEnd>intStart)
   {
      strTitle = StringSubstr(strMiddle,intStart+7,intEnd-intStart-7);
      strTitle=StringTrimLeft(StringTrimRight(strTitle));
   }

   //extract the country.
   intStart=StringFind(strMiddle,"<COUNTRY>",0);
   intEnd = StringFind(strMiddle,"</COUNTRY>",intStart);
   if(intStart>=0 && intEnd>intStart)
   {
      strCountry = StringSubstr(strMiddle,intStart+9,intEnd-intStart-9);
      strCountry=StringTrimLeft(StringTrimRight(strCountry));
   }

   //extract the date.
   intStart=StringFind(strMiddle,"<DATE>",0);
   intEnd = StringFind(strMiddle,"</DATE>",intStart);
   if(intStart>=0 && intEnd>intStart)
   {
      strDate = StringSubstr(strMiddle,intStart+6,intEnd-intStart-6);
      //We should have something like <![CDATA [ 12-04-2016 ]]>
      //remove everything up to CDATA
      intStart=StringFind(strDate,"CDATA",0);
      strDate=StringSubstr(strDate,intStart+5,StringLen(strDate)-intStart-5);
//should have [ 12-04-2016 ]]>
      //remove everything before the bracket, [ after CDATA.
      intStart=StringFind(strDate,"[",0);
      strDate=StringSubstr(strDate,intStart+1,StringLen(strDate)-intStart-1);
//should have 12-04-2016 ]]>
      //remove everything before the first ]
      intEnd = StringFind(strDate,"]",0);
      strDate = StringSubstr(strDate,0,intEnd); //should have 12-04-2016
      //Finally trim the result
      strDate = StringTrimRight(StringTrimLeft(strDate));
      //Remove any white space
```

```
      StringReplace(strDate," ","");

//Now we should have something like 12-04-2016.   Extract the month first, which is the firs
      intStart = StringFind(strDate,"-",0);
      if(intStart<=0)   intStart=StringFind(strDate,".",0);
      strMonth=StringSubstr(strDate,0,intStart);   //should have 12
      if(StringLen(strMonth)<2)  strMonth="0"+strMonth;
//make sure format is 05 instead of 5
      //Now extract the day
      intEnd=StringFind(strDate,"-",intStart+1);
      if(intEnd<=0)   intEnd=StringFind(strDate,".",intStart+1);
      strDay=StringSubstr(strDate,intStart+1,intEnd-intStart-1);  //should have 04
      if(StringLen(strDay)<2) strDay="0"+strDay;   //make sure format is 04 instead of 4
      //Get the year
      strYear=StringSubstr(strDate,intEnd+1,StringLen(strDate)-intEnd-1);
   }
   //extract the time.
   intStart=StringFind(strMiddle,"<TIME>",0);
   intEnd = StringFind(strMiddle,"</TIME>",intStart);
   if(intStart>=0 && intEnd>intStart)
   {
      strTime = StringSubstr(strMiddle,intStart+7,intEnd-intStart-7);
      //We should have something like <![CDATA [ 11:20am ]]>
      //remove everything up to CDATA
      intStart=StringFind(strTime,"CDATA",0);
      strTime=StringSubstr(strTime,intStart+5,StringLen(strTime)-intStart-5);
//should have [ 11:20am ]]>
      //remove everything before the bracket, [ after CDATA.
      intStart=StringFind(strTime,"[",0);
      strTime=StringSubstr(strTime,intStart+1,StringLen(strTime)-intStart-1);
//should have 11:20am ]]>
      //remove everything before the first ]
      intEnd = StringFind(strTime,"]",0);
      strTime = StringSubstr(strTime,0,intEnd); //should have 11:20am
      //Finally trim the result
      strTime = StringTrimRight(StringTrimLeft(strTime));
      //Remove any white space
      StringReplace(strTime," ","");

//Now we should have something like "11:20am" or possibly 11:20.   Extract the am or pm firs
      strAmPm="";
      if(StringLen(strTime)>4)   //eg 1:20am or 11:00pm are longer than 4
      {
         strAmPm=StringSubstr(strTime,StringLen(strTime)-2,2);
         StringToUpper(strAmPm);
         if(strAmPm=="AM" || strAmPm=="PM")
         {
            strTime=StringSubstr(strTime,0,StringLen(strTime)-2);
         }
      }
      //Get the hour
      intEnd=StringFind(strTime,":",0);
      if(intEnd<=0)   intEnd=StringFind(strTime,".",0);
      strHour=StringSubstr(strTime,0,intEnd);   //should have 11
      //Get the mins
      strMin=StringSubstr(strTime,intEnd+1,StringLen(strTime)-intEnd-1);
//should have 20
   }

   //extract the impact.
   intStart=StringFind(strMiddle,"<IMPACT>",0);
   intEnd = StringFind(strMiddle,"</IMPACT>",intStart);
   if(intStart>=0 && intEnd>intStart)
   {
      strImpact = StringSubstr(strMiddle,intStart+7,intEnd-intStart-7);
      //We should have something like <![CDATA [ High ]]>
      //remove everything up to CDATA
      intStart=StringFind(strImpact,"CDATA",0);
      strImpact=StringSubstr(strImpact,intStart+5,StringLen(strImpact)-intStart-5);
//should have "[ High ]]>"
      //remove everything before the bracket, [ after CDATA.
```

```
      intStart=StringFind(strImpact,"[",0);
      strImpact=StringSubstr(strImpact,intStart+1,StringLen(strImpact)-intStart-1);
//should have " High ]]>"
      //remove everything before the first ]
      intEnd = StringFind(strImpact,"]",0);
      strImpact = StringSubstr(strImpact,0,intEnd);   //should have " High "
      //Finally trim the result
      strImpact = StringTrimRight(StringTrimLeft(strImpact));  //should have "High"
      //we should have something like "High", "Low" etc.
      //Get the first 3 charaters
      strImpact = StringSubstr(strImpact,0,3);  //should have Hig, Med, Low or Hol
      //Set the impact
      StringToUpper(strImpact);
      if (strImpact=="HIG")
      {
         eImpact=HighImpact;
      }
      else if(strImpact=="MED")
      {
         eImpact=MediumImpact;
      }
      else if(strImpact=="HOL")
      {
         eImpact=Holiday;
      }
      else
      {
         eImpact=LowImpact;
      }
   }

   //Process the date and time
   if(this.StringContainsNumbersOnly(strYear) && this.StringContainsNumbersOnly(strMonth
) && this.StringContainsNumbersOnly(strDay) && this.StringContainsNumbersOnly(strHour)
&& this.StringContainsNumbersOnly(strMin))
   {


//First, convert the hour and minute into integers, evaluate am and pm and then work out the
      int intHour = (int)StringToInteger(strHour);
      int intMin = (int)StringToInteger(strMin);
      if(intHour==12)
      {
         intHour=0;
      }
      if(strAmPm=="PM")
      {
         intHour+=12;
      }
      strHour=IntegerToString(intHour);
      strMin=IntegerToString(intMin);

      //Make sure the hour and minute are sensible, otherwise exit.
      if(intHour>23 || intHour<0 || intMin>59 || intMin<0)  return False;

      //Make sure that the year, month and day are reasonable
      int intDay = (int)StringToInteger(strDay);
      int intMonth = (int)StringToInteger(strMonth);
      int intYear = (int)StringToInteger(strYear);
      if(intDay<1 || intDay>31 || intMonth<1 || intMonth>12 || intYear<0 || intYear>3000
)   return False;

      //Make sure there are at least two characters in the hour and minute format.
      if(StringLen(strHour)<2)   strHour="0"+strHour;
      if(StringLen(strMin)<2) strMin="0"+strMin;

      //Convert the time into an hh:mm format
      strTime = strHour+":"+strMin;

      //Make sure there are at least two characters in the month and day format.
      if(StringLen(strMonth)<2)  strMonth="0"+strMonth;
```

```
      if(StringLen(strDay)<2) strDay="0"+strDay;

      //Next create a datetime format
      string strDateTimeFormat = strYear+"."+strMonth+"."+strDay+" "+strTime;
      datetime dtTimeResult = StringToTime(strDateTimeFormat);

      //Now apply the calendar offset to the datetime
      dtTimeResult+=this.m_intCalendarTimeOffsetSecs;

      //Finally populate the array with the new data
      C_Calendar_Data *objCD = new C_Calendar_Data(dtTimeResult,strCountry,strTitle,
eImpact);
      if(objCD.IsBankHoliday())
      {
         this.AddBankHoliday(objCD);
      }
      else
      {
         //If the impact level is valid and country list is valid, then import it.
         if(this.ImpactIsValidForImport(eImpact) && this.CountryIsValidForImport(
strCountry))   this.AddCalendarData(objCD);
      }
      delete objCD;

      return True;

   }
   else if(this.StringContainsNumbersOnly(strYear) && this.StringContainsNumbersOnly(
strMonth) && this.StringContainsNumbersOnly(strDay))
   {  //We have a date, but the time is not numeric.   It is an all day event.

      string strTimeTemp=strTime;
      StringToUpper(strTimeTemp);

      if((StringFind(strTimeTemp,"ALL",0)>=0) && (StringFind(strTimeTemp,"DAY",0)>=0))
      {

         //Make sure that the year, month and day are reasonable
         int intDay = (int)StringToInteger(strDay);
         int intMonth = (int)StringToInteger(strMonth);
         int intYear = (int)StringToInteger(strYear);
         if(intDay<1 || intDay>31 || intMonth<1 || intMonth>12 || intYear<0 || intYear>
3000)   return False;


         //Make sure there are at least two characters in the month and day format.
         if(StringLen(strMonth)<2)  strMonth="0"+strMonth;
         if(StringLen(strDay)<2) strDay="0"+strDay;

         //Next create a datetime format
         string strDateTimeFormat = strYear+"."+strMonth+"."+strDay+" 00:00";
         datetime dtTimeResult = StringToTime(strDateTimeFormat);


//DO NOT APPLY A TIME OFFSET TO ALL DAY EVENTS.   I DON'T SEE HOW THIS CAN WORK.

         //Finally populate the array with the new data
         C_Calendar_Data *objCD = new C_Calendar_Data(dtTimeResult,strCountry,strTitle,
eImpact);

         //If the impact level is valid and country list is valid, then import it.
         if(this.ImpactIsValidForImport(eImpact) && this.CountryIsValidForImport(
strCountry))   this.AddAllDayEvent(objCD);

         delete objCD;

         return True;
      }
   }

   return False;
```

```
}
bool C_Calendar_ForexFactoryImport::StringContainsNumbersOnly(const string strInput)
{
   //Fill an array with the string char's.
   bool boolRet=True;
   uchar ucArray[];
   StringToCharArray(strInput,ucArray,0,WHOLE_ARRAY,CP_ACP);

   //Scan through the array to see if it contains any non numeric characters.
   for(int i=0;i<StringLen(strInput);i++)
   {
      if(ucArray[i]<48 || ucArray[i]>57)
      {
         boolRet=False;
         break;
      }
   }

   return boolRet;

}
bool C_Calendar_ForexFactoryImport::CountryIsValidForImport(const string strCountry)
{
   bool boolRet=False;
   int intCount=ArraySize(this.m_strImportCountryList);
   if(intCount<1) return boolRet;
   for(int i=0;i<intCount;i++)
   {
      string strCUT=StringTrimRight(StringTrimLeft(strCountry));
      StringToUpper(strCUT);
      string strIUT=StringTrimRight(StringTrimLeft(this.m_strImportCountryList[i]));
      StringToUpper(strIUT);
      if(strCUT==strIUT)
      {
         boolRet=True;
         break;
      }
   }
   return boolRet;
}
bool C_Calendar_ForexFactoryImport::ImpactIsValidForImport(const NewsImpact eImpact)
{
   bool boolRet=False;
   if(this.m_boolImportLowImpact && eImpact==LowImpact)   boolRet=True;
   if(this.m_boolImportMediumImpact && eImpact==MediumImpact)   boolRet=True;
   if(this.m_boolImportHighImpact && eImpact==HighImpact)   boolRet=True;
   return boolRet;
}
```