Test agile

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace tstBooking\_cs

{

[TestClass]

public class tstBooking

{

string BookingId = "1";

string SessionId = "2";

string BookingStatus = "Completed";

string MemberID = "4";

string BookingDateTime = DateTime.Now.Date.ToString();

public object ABooking { get; private set; }

public int TestBookingID { get; private set; }

[TestMethod]

public void InstanceOK()

{

clsBooking ABooking = new clsBooking();

Assert.IsNotNull(ABooking);

}

[TestMethod]

public void ActivePropertyOK()

{

//create an instance of the class we weant to create

clsBooking ABooking = new clsBooking();

//create the test data to the property

Boolean TestData = true;

//assign the data to the property

ABooking.Active = TestData;

//test to see that the two values are the same

Assert.AreEqual(ABooking.Active, TestData);

}

[TestMethod]

public void DateTimePropertyOK()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//create some test data to assign to the property

DateTime TestData = DateTime.Now.Date;

//assign the data to the property

ABooking.DateTime = TestData;

//test to see that the two values are the same

Assert.AreEqual(ABooking.DateTime, TestData);

}

[TestMethod]

public void BookingIDPropertyOK()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//create some test data to assign to the property

int testID = 2;

//assign the data to the property

ABooking.BookingID = testID;

//test to see that the two values are the same

Assert.AreEqual(ABooking.BookingID, testID);

}

[TestMethod]

public void SessionIDPropertyOK()

{

clsSession ASession = new clsSession();

int TestID = 2;

ASession.SessionID = TestID;

Assert.AreEqual(ASession.SessionID, TestID);

}

[TestMethod]

public void MemberIDPropertyOK()

{

clsMember AMember = new clsMember();

int TestID = 4;

AMember.MemberID = TestID;

Assert.AreEqual(AMember.MemberID, TestID);

}

[TestMethod]

public void BookingStatusOK()

{

clsBooking ABooking = new clsBooking();

string status = "Completed";

ABooking.BookingStatus = status;

Assert.AreEqual(ABooking.BookingStatus, status);

}

[TestMethod]

public void FindMethodOK()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//boolean variable to store the result of the validaiton

Boolean Found = false;

//create some test data to use with the method

int TestBookingID = 2;

//invoke the method

Found = ABooking.Find(TestBookingID);

Assert.IsTrue(Found);

}

[TestMethod]

public void TestBookingIDFound()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

int32 TestBookingID = 2;

//invoke the method

Found = ABooking.Find(TestBookingID);

//check the Booking ID

if (ABooking.BookingID != 2)

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestDateTimeFound()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 TestBookingID = 2;

//invoke the method

Found = ABooking.Find(TestBookingID);

//check the property

if (ABooking.DateTime != Convert.ToDateTime("22/01/2019 12:00:00"))

{

OK = false;

}

//test ot see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestBookingStatusFound()

{

clsBooking ABooking = new clsBooking();

Boolean Found = false;

Boolean OK = true;

Int32 TestBookingID = 2;

Found = ABooking.Find(TestBookingID);

if (ABooking.Status != "false")

{

OK = false;

}

Assert.IsTrue(OK);

}

[TestMethod]

public void ValidMethodOK()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

//test to see that hte result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDMinLessOne()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string SessionID = ""; //should fail

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

//test to see that the result is correct

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void SessionIDMin()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "A";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDMinPlusOne()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "A A";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDMaxLessOne()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionId = "AA A";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDMax()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "AAAAAA";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus,SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIdMid()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "AAAAAAA";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDMaxPlusOne()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "AAAAAAA";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void SessionIDExtremeMax()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string SessionID = "";

SessionID = SessionID.PadRight(100, 'A') + " ";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus,SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDMinLessOne()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string BranchID = "";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void BranchIDMin()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string BranchID = "";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDMaxLessOne()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string BranchID = "aaaaa"; //this should be ok

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDMax()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string BranchID = "aaaaaa"; //this should be ok

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDMid()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string BranchID = "aaa"; //this should be ok

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDMaxPlusOne()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string BranchID = "aaaaaa";

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, BranchID, SessionId, MemberID);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BranchIDExtremeMax()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create some test data to pass to the method

string BranchID = "";

BranchID = BranchID.PadRight(100, 'a') + " ";

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus,SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void DateTimeExtremeMin()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create a variable to store the test date data

DateTime TestDate;

//set the date totodays date

TestDate = DateTime.Now.Date;

//change the date to whatever the date is less 100 years

TestDate = TestDate.AddYears(-100);

//convert the date variable to a string variable

string dateTime = TestDate.ToString();

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void DateTimeMinLessOne()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create a variable to store the test date data

DateTime TestDate;

//set the date totodays date

TestDate = DateTime.Now.Date;

//change the date to whatever the date is less 1 day

TestDate = TestDate.AddDays(-1);

//convert the date variable to a string variable

string dateTime = TestDate.ToString();

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void DateTimeMin()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create a variable to store the test date data

DateTime TestDate;

//set the date totodays date

TestDate = DateTime.Now.Date;

//convert the date variable to a string variable

string DateAdded = TestDate.ToString();

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void DateTimeExtremeMax()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//create a variable to store the test date data

DateTime TestDate;

//set the date totodays date

TestDate = DateTime.Now.Date;

//change the date to whatever the date is plus 100 years

TestDate = TestDate.AddYears(100);

//convert the date variable to a string variable

string dateTime = TestDate.ToString();

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void DateTimeInvalidData()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

//set the DateTie to a non date value

string DateTime = "this is not a date!";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void BookingStatusMinLessOne()

{

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

string Status = "";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void BookingStatusMin()

{

//create an instance of the class we want to create

clsBooking ASession = new clsBooking();

//string variable to store any error message

string Error = "";

string status = " Completed";

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BookingDateTimeMin()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

String Error = "";

//create a variable to store the test date data

DateTime TestDate;

//set the date totodays date

TestDate = DateTime.Now.Date;

//convert the date variable to a string variable

string DateAdded = TestDate.ToString();

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BookingStatusMinPlusOne()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

string status = "";

//invoke the method

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

//test to see that the result is correct

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BookingStatusMaxLessone()

{

//create an instance of the class we want to create

clsBooking ABooking = new clsBooking();

//string variable to store any error message

string Error = "";

string status = "";

status = status.PadRight(49, 'a');

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BookingStatusMax()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string status = "";

status = status.PadRight(50, 'a');

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

Assert.AreEqual(Error, "");

}

[TestMethod]

public void BookingStatusMaxPlusOne()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string Status = "Completed";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

Assert.AreNotEqual(Error, "");

}

[TestMethod]

public void BookingStatusMid()

{

clsBooking ABooking = new clsBooking();

string Error = "";

string Status = "Completed ";

Error = ABooking.Valid(BookingId, BookingDateTime, BookingStatus, MemberID, BookingStatus, SessionId);

Assert.AreEqual(Error, "");

}

}

internal class clsMember

{

public clsMember()

{

}

public int MemberID { get; internal set; }

}

internal class clsSession

{

public clsSession()

{

}

public int SessionID { get; internal set; }

}

internal class clsBooking

{

public clsBooking()

{

}

public bool Active { get; internal set; }

public DateTime DateTime { get; internal set; }

public int BookingID { get; internal set; }

public string BookingStatus { get; internal set; }

public string Status { get; internal set; }

internal bool Find(int testBookingID)

{

throw new NotImplementedException();

}

internal string Valid(string bookingId, string bookingDateTime, string bookingStatus, object branchID, string sessionId, string memberID)

{

throw new NotImplementedException();

}

internal string Valid(string bookingId, string bookingDateTime, string bookingStatus, string sessionId, string memberID)

{

throw new NotImplementedException();

}

}

}