using System;

using System.Threading.Tasks;

using Pecunia.Entities;

using Pecunia.BusinessLayer;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Pecunia.UnitTest

{

[TestClass]

public class ChequeBookBLTest

{

[TestMethod]

public async Task AddValidChequeBook()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000000", SeriesStart = 200000, NumberOfLeaves = 10 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsTrue(isAdded, errorMessage);

}

}

/// <summary>

/// Accountno can't be null

/// </summary>

[TestMethod]

public async Task ChequeBookAccountNoCanNotBeNull()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = null, SeriesStart = 200000, NumberOfLeaves = 20 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Series can't be null

/// </summary>

[TestMethod]

public async Task ChequeBookSeriesStartCanNotBeNull()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000000", SeriesStart = 0, NumberOfLeaves = 20 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Number of leaves can't be null

/// </summary>

[TestMethod]

public async Task ChequeBookNumberOfLeavesCanNotBeNull()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000000", SeriesStart = 100000, NumberOfLeaves = 0 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Account number should contain 10 digits

/// </summary>

[TestMethod]

public async Task ChequeBookAccountNumberRegExp()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "100000", SeriesStart = 100000, NumberOfLeaves = 10 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Series start number should contain 6 digits

/// </summary>

[TestMethod]

public async Task ChequeBookSeriesStartRegExp()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000000", SeriesStart = 1000, NumberOfLeaves = 20 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Number of leaves should be multiples of 5

/// </summary>

[TestMethod]

public async Task ChequeBookNumberOfLeavesShouldBeValid()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000000", SeriesStart = 200000, NumberOfLeaves = 28 };

bool isAdded = false;

string errorMessage = null;

//Act

try

{

isAdded = await chequeBookBL.AddChequeBookBL(chequeBook);

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Get ChequeBook if ChequeBookId is valid

/// </summary>

[TestMethod]

public async Task ValidChequeBookByChequeBookId()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000001", SeriesStart = 300000, NumberOfLeaves = 20 };

await chequeBookBL.AddChequeBookBL(chequeBook);

bool isAdded = false;

string errorMessage = null;

//Act

try

{

if (chequeBook.Equals(await chequeBookBL.GetChequeBookByChequeBookIdBL(chequeBook.ChequeBookId)))

{ isAdded = true; }

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsTrue(isAdded, errorMessage);

}

}

/// <summary>

/// Show error ChequeBook Id is invalid

/// </summary>

[TestMethod]

public async Task InValidChequeBookByChequeBookId()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000002", SeriesStart = 100000, NumberOfLeaves = 20 };

await chequeBookBL.AddChequeBookBL(chequeBook);

Guid chequeBookID = new Guid();

bool isAdded = true;

string errorMessage = null;

//Act

try

{

if ((chequeBook = (await chequeBookBL.GetChequeBookByChequeBookIdBL(chequeBookID))) == null)

{ isAdded = false; }

}

catch (Exception ex)

{

isAdded = true;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

/// <summary>

/// Get ChequeBook if Series start is valid

/// </summary>

[TestMethod]

public async Task ValidChequeBookBySeriesStart()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000003", SeriesStart = 500000, NumberOfLeaves = 20 };

await chequeBookBL.AddChequeBookBL(chequeBook);

bool isAdded = false;

string errorMessage = null;

//Act

try

{

if (chequeBook.Equals(await chequeBookBL.GetChequeBookBySeriesStartBL(chequeBook.SeriesStart)))

{ isAdded = true; }

}

catch (Exception ex)

{

isAdded = false;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsTrue(isAdded, errorMessage);

}

}

/// <summary>

///ShoW error if Series start is invalid

/// </summary>

[TestMethod]

public async Task InValidSeriesStartForChequeBook()

{

//Arrange

ChequeBookBL chequeBookBL = new ChequeBookBL();

ChequeBook chequeBook = new ChequeBook() { AccountNo = "1000000004", SeriesStart = 400000, NumberOfLeaves = 20 };

await chequeBookBL.AddChequeBookBL(chequeBook);

double seriesStart = 00;

bool isAdded = true;

string errorMessage = null;

//Act

try

{

if ((chequeBook = (await chequeBookBL.GetChequeBookBySeriesStartBL(seriesStart))) == null)

{ isAdded = false; }

}

catch (Exception ex)

{

isAdded = true;

errorMessage = ex.Message;

}

finally

{

//Assert

Assert.IsFalse(isAdded, errorMessage);

}

}

}

}