## DCUreg

# DUBLIN CITY UNIVERSITY

**SEMESTER 2** **SOLUTIONS** **2017/2018**

**MODULE:** EE417 - Web Application Development

### PROGRAMME(S):

|  |  |
| --- | --- |
| ECE | BEng Electronic & Computer Engineering |
| MECE | MEng Electronic & Computer Engineering |
| ECSAO | Study Abroad (Engineering & Computing) |
| MTC | MEng in Telecommunications Engineering |
| MEQ | Masters Engineering Qualifier Course |
|  |  |

**YEAR OF STUDY:** 4,C,O

**EXAMINER(S):**

|  |  |
| --- | --- |
| David Molloy | (Internal) |
| Dr. Iain Phillips | (External) |
| Prof. Gerard Parr | (External) |

**TIME ALLOWED:** 3 Hours

**INSTRUCTIONS: Answer Question 1**

**Answer Three Questions in Total**

**PLEASE DO NOT TURN OVER THIS PAGE UNTIL YOU ARE INSTRUCTED TO DO SO.**

The use of programmable or text storing calculators is expressly forbidden.

Please note that where a candidate answers more than the required number of optional questions, the examiner will mark all optional questions attempted and then select the highest scoring ones.

***There are no additional requirements for this paper.***

## QUESTION 1 (Solutions) [TOTAL MARKS: 50]

### Q 1(a) [11 Marks]

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Calculator</title>

<style>

*.bodyStyle* {width:*450px*}

*.boxStyle* { float:*left*;width:*100px*;height:*100px*;border:*1px solid black*;margin:*24px*;}

*.textStyle* { text-align:*center*;padding-top:*40px*;font-size:*20pt*;}

*.filled* { background-color:*lightgrey*;}

*.inputstyle* { width:*50px*;margin:*20px*;margin-top:*30px*;font-size:*20pt*;}

*.submitStyle* { margin:*20px*}

</style>

</head>

<body class=*"bodyStyle"*>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>1</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>2</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>3</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>4</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>5</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>6</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>7</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>8</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>9</div></div>

<div class=*"boxStyle filled"*><div class=*"textStyle"*>0</div></div>

<form>

<div class=*"boxStyle"*><input type=*"text"* name=*"valuea"* class=*"inputstyle"* value=*""*/></div>

<div class=*"boxStyle"*><input type=*"text"* name=*"valueb"* class=*"inputstyle"* value=*""*/></div>

<div style="margin-left:*20px*">JavaScript Result = <span id=*"jsSum"*></span></div>

</form>

</body>

</html>

### Q 1(b) [10 Marks]

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Calculator</title>

<style>

*.bodyStyle* {width:*450px*;font-size:*15pt*;}

*.boxStyle* { float:*left*;width:*100px*;height:*100px*;border:*1px solid black*;margin:*24px*;}

*.textStyle* { text-align:*center*;padding-top:*40px*;font-size:*20pt*;}

*.filled* { background-color:*lightgrey*;}

*.inputstyle* { width:*50px*;margin:*20px*;margin-top:*30px*;font-size:*20pt*;}

*.submitStyle* { margin:*20px*}

</style>

<script>

**var** selected = "valuea";

**function** selectInputA() {

selected = "valuea";

}

**function** selectInputB() {

selected = "valueb";

}

**function** setValue(value) {

**if** (selected=="valueb")

document.myform.valueb.value = value;

**else**

document.myform.valuea.value = value;

document.getElementById("jsSum").innerHTML = document.myform.valueb.value \* document.myform.valuea.value;

}

</script>

</head>

<body class=*"bodyStyle"*>

<div class=*"boxStyle filled"* onClick="setValue(1)"><div class=*"textStyle"*>1</div></div>

<div class=*"boxStyle filled"* onClick="setValue(2)"><div class=*"textStyle"*>2</div></div>

<div class=*"boxStyle filled"* onClick="setValue(3)"><div class=*"textStyle"*>3</div></div>

<div class=*"boxStyle filled"* onClick="setValue(4)"><div class=*"textStyle"*>4</div></div>

<div class=*"boxStyle filled"* onClick="setValue(5)"><div class=*"textStyle"*>5</div></div>

<div class=*"boxStyle filled"* onClick="setValue(6)"><div class=*"textStyle"*>6</div></div>

<div class=*"boxStyle filled"* onClick="setValue(7)"><div class=*"textStyle"*>7</div></div>

<div class=*"boxStyle filled"* onClick="setValue(8)"><div class=*"textStyle"*>8</div></div>

<div class=*"boxStyle filled"* onClick="setValue(9)"><div class=*"textStyle"*>9</div></div>

<div class=*"boxStyle filled"* onClick="setValue(0)"><div class=*"textStyle"*>0</div></div>

<form name=*"myform"*>

<div class=*"boxStyle"* onclick="selectInputA()"><input type=*"text"* onclick="selectInputA()" name=*"valuea"* class=*"inputstyle"* value=*""*/></div>

<div class=*"boxStyle"* onclick="selectInputB()"><input type=*"text"* onclick="selectInputB()" name=*"valueb"* class=*"inputstyle"* value=*""*/></div>

<div style="margin-left:*20px*">JavaScript Result = <span id=*"jsSum"*></span></div>

</form>

</body>

### </html>

### Q 1(c) [12 Marks]

**(9 Marks for Servlet)**

**package** org.ee417;

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**import** javax.servlet.annotation.WebServlet;

**import** org.ee417.beans.Calculation;

@WebServlet("/Calculator")

**public** **class** Calculator **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **void** doGet(HttpServletRequest req, HttpServletResponse res)

**throws** ServletException, IOException {

res.setContentType("text/html");

PrintWriter out = res.getWriter();

out.println("<html><head><title>Calculator Output</title></head>");

out.println("<body><h1>Server-side Calculator</h1>");

**try** {

**int** val1 = **new** Integer(req.getParameter("valuea")).intValue();

**int** val2 = **new** Integer(req.getParameter("valueb")).intValue();

Calculation calculation = **new** Calculation(val1, val2);

out.println("This Calculation: " + calculation.getA() + " + " + calculation.getB() + " = " + calculation.getResult());

HttpSession session = req.getSession();

Calculation calc = (Calculation) session.getAttribute("myCalc");

**if** (calc!=**null**) out.println("<br/>Previous Calculation: " + calc.getA() + " \* " + calc.getB()+ " = " + calc.getResult());

session.setAttribute("myCalc", calculation);

} **catch** (Exception e) {

out.println("Invalid values submitted. Please provide two valid int values!");

}

out.println("</body></html>");

out.close();

}

### }

**(3 Marks for JavaBean)**

**package** org.ee417.beans;

**public** **class** Calculation {

**int** a;

**int** b;

**int** result;

**public** Calculation(**int** a, **int** b) {

**super**();

**this**.a = a;

**this**.b = b;

**this**.result = a \* b;

}

**public** **int** getA() {

**return** a;

}

**public** **void** setA(**int** a) {

**this**.a = a;

}

**public** **int** getB() {

**return** b;

}

**public** **void** setB(**int** b) {

**this**.b = b;

}

**public** **int** getResult() {

**return** result;

}

**public** **void** setResult(**int** result) {

**this**.result = result;

}

}

### Q 1(d) [8 Marks]

package org.ee417;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

import org.ee417.beans.Calculation;

public class MyTest {

@Test

public void testMultiplyNormal() {

System.out.println("MyTest: testMultiplyNormal2()");

Calculation c = new Calculation(3,9);

assertEquals(27,c.getResult());

}

@Test

public void testMultiplyMinus() {

System.out.println("MyTest: testMultiplyMinus()");

Calculation c = new Calculation(-1,6);

assertEquals(-6,c.getResult());

}

@Test

public void testMultiplyMaxIntRange() {

System.out.println("MyTest: testMultiplyMaxIntRange()");

Calculation c = new Calculation(2147483647,200);

assertEquals(0,c.getResult());

}

}

### Q 1(e) [3 Marks]

Discussion about how the calculation could be shown directly on the same HTML page using in-page replacement. This would make the application work more like a modern single-page application rather than a series of forms.

### Q 1(e) [6 Marks]

Any code example, which shows a declaration, scriptlet and expression. For example, the following three could be used.

<%! String testString = "Hello World!"; %>

<% out.println("Your address is: " + request.getRemoteAddr()); %>

The current time is: <%= new java.util.Date() %>

## [End of Solutions of Question 1]

## 

## QUESTION 2 (Solutions) [TOTAL MARKS: 25]

### Q 2(a) [12 Marks]

CREATE TABLE CLIENTS (2 Marks)

(

ID INTEGER,

USERNAME VARCHAR(30) NOT NULL,

PASSWORD VARCHAR(30) NOT NULL,

EMAIL VARCHAR(150) NOT NULL,

PRIMARY KEY(ID)

)

CREATE TABLE PROFILES (3 Marks)

(

P\_ID INTEGER,

NAME VARCHAR(50) NOT NULL,

SEEKING VARCHAR(30),

AGE INTEGER,

DESCRIPTION VARCHAR(30),

C\_ID INTEGER,

PRIMARY KEY(P\_ID),

CONSTRAINT CNST1

FOREIGN KEY (C\_ID)

REFERENCES CLIENTS

ON DELETE CASCADE

)

CREATE TABLE DATES (4 Marks)

(

D\_ID INTEGER,

PERSON1 INTEGER,

PERSON2 INTEGER,

DATE\_TIME VARCHAR(30),

DATE\_LOC VARCHAR(100),

PRIMARY KEY(D\_ID),

CONSTRAINT CNST2

FOREIGN KEY (PERSON1)

REFERENCES CLIENTS

ON DELETE SET NULL,

CONSTRAINT CNST3

FOREIGN KEY (PERSON2)

REFERENCES CLIENTS

ON DELETE SET NULL

)

(3 Marks)

INSERT INTO CLIENTS VALUES (1, “smithj”, “somepass”, “smithj@google.com”)

INSERT INTO CLIENTS VALUES (2, “doej”, “anotherpass”, “doej@gmail.com”)

INSERT INTO PROFILES VALUES

( 101, “John Smith”, “Women”, 24, “Loves dogs”, 1)

INSERT INTO PROFILES VALUES

( 102, “Jane Doe”, “Men”, 25, “Loves cats”, 2)

INSERT INTO DATES VALUES

( 1001, 1, 2, ‘25-03-13’, ‘El Grigorio Restaurant’)

### 

### Q 2(b) [4 Marks]

Create index ageIndex on profiles(age)

In this scenario we are adding an index to the age column of the PROFILES table. In a dating system it should be common that people will search for other users commonly using the ‘Age’ field. This index will speed up searches involving queries against age.

### Q 2(c) [5 Marks]

The web service client proxy is written so that it appears as though the code is being executed locally and looks very much like the creation of a local constructor. However, in reality what is happening is that XML is being used for communication between the local client and the remote web service. This communication is happening probably using SOAP which handles the sending and receiving of these XML messages.

Hence, in this scenario, the StringToolsProxy.upperCase(“Testing”) request actually creates an XML message with the parameters to be passed over HTTP to the web service. The remote web service performs the operations, returns the result in XML and the client parses this XML to get the result.

### Q 2(d) [4 Marks]

GORM (Grails Object Relational Mapping) is Grails’ Object Relational Mapper. It improves the convention and reduces the amount of traditional Hibernate code required to interact with domain objects in a database.

Grails supports the idea of dynamic finders. These look like methods that have been written, except that code is never manually written. Rather, from the name of these dynamic methods, Grails can understand what is being requested and dynamically process the requests. Eg. def list = User.findAllBySurname(“Smith”);

## [End of Solutions of Question 2]

## QUESTION 3 (Solutions) [TOTAL MARKS: 25]

### Q 3(a) [5 Marks]

BankAccount.java

**package** test;

**import** javax.persistence.\*;

@Entity

@Table (name="BankAccountTable")

**public** **class** BankAccount {

**private** **int** id;

**private** String accountNumber;

**private** String sortCode;

**private** String accountType;

**private** **double** balance;

**private** int bankCustomer;

**public** BankAccount(**int** id, String accountNumber, String sortCode,

String accountType, **double** balance, int bankCustomer) {

**super**();

**this**.id = id;

**this**.accountNumber = accountNumber;

**this**.sortCode = sortCode;

**this**.accountType = accountType;

**this**.balance = balance;

this.bankCustomer = bankCustomer;

}

**public** BankAccount() {

**super**();

}

@Id

@GeneratedValue

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getAccountNumber() {

**return** accountNumber;

}

**public** **void** setAccountNumber(String accountNumber) {

**this**.accountNumber = accountNumber;

}

**public** String getSortCode() {

**return** sortCode;

}

**public** **void** setSortCode(String sortCode) {

**this**.sortCode = sortCode;

}

**public** String getAccountType() {

**return** accountType;

}

**public** **void** setAccountType(String accountType) {

**this**.accountType = accountType;

}

**public** **double** getBalance() {

**return** balance;

}

**public** **void** setBalance(**double** balance) {

**this**.balance = balance;

}

**public** int getBankCustomer() {

**return** bankCustomer;

}

**public** **void** setBankCustomer(int bankCustomer) {

**this**.bankCustomer = bankCustomer;

}

}

### Q 3(b) [7 Marks]

package edu.ee.hibernate;

import edu.ee.beans.BankAccount;

import edu.ee.hibernate.HibernateUtil;

import org.hibernate.Session;

public class MyBank {

public static void main(String[] args) {

Session session = HibernateUtil.beginTransaction();

HibernateUtil.recreateDatabase();

BankAccount bankAccount1 =

new BankAccount(2, "12345678", "901122", "Savings", 4533.92, 1);

BankAccount bankAccount2 =

new BankAccount(3, "98765432", "901122", "Current", 255.78, 1);

BankAccount bankAccount3 =

new BankAccount(5, "55555555", "901187", "Savings", 4533.92, 4);

BankAccount bankAccount4 =

new BankAccount(6, "66666666", "901144", "Deposit", 443.13, 4);

session.save(bankAccount1);

session.save(bankAccount2);

session.save(bankAccount3);

session.save(bankAccount4);

bankAccount1.setBalance(1122.33);

session.update(bankAccount1);

session.delete(bankAccount4);

HibernateUtil.commitTransaction();

HibernateUtil.closeSession();

}

}

### Q 3(c) [5 Marks]

* The first line defines the JDBC Url required for connecting to the database. It defines the database address, the port, driver type and the SSID of the database we wish to connect to.
* We then load the JDBC driver into the JVM for later use for when we later try to open a connection to the database.
* Next we open a connection to the database using the JDBC url, the database username and password and the JDBC Driver which was loaded into the JVM previously.
* We set the connection commit mode to false, which means that we will need to explicitly commit our code at the end of the transaction
* We use the connection to create a new Statement object
* We execute a statement telling the database to drop the table called ‘mytable’

### Q 3(d) [4 Marks]

**Database Security** – one of the issues associated with providing a database in a two-tier system is that the database needs to be accessible from the location of every possible user of the system. This means that the database is unnecessarily exposed and can not be effectively firewalled. Three-tier systems, on the other hand, tend to hide their databases behind a server firewall, so that only the application server (and some local DBA machines) are allowed access.

**Code Protection –** on two tier systems, the code typically takes the form of a client-side application, such as an executable (.exe) file or a Java (.class) applet. For this to work, the binary version of the code must be downloaded to every user. While this is not a problem in itself, it means that there are many binary copies of the file in existence and malicious users may attempt to decompile the code to steal it or to discover vulnerabilities or backdoors. On a three-tier system this cannot occur as the server code is never downloaded to the client.

### Q 3(e) [4 Marks]

**Accessibility** – Web accessibility refers to the inclusive practice of removing barriers that prevent interaction with, or access to websites, by people with disabilities. When sites are correctly designed, developed and edited, all users have equal access to information and functionality.

**Responsiveness –** Responsive web design is an approach to web design which makes web pages render well on a variety of devices and window or screen sizes. This has become increasingly important with the advent of widespread usage of handheld devices as well as continued usage of large screen devices such as PCs.

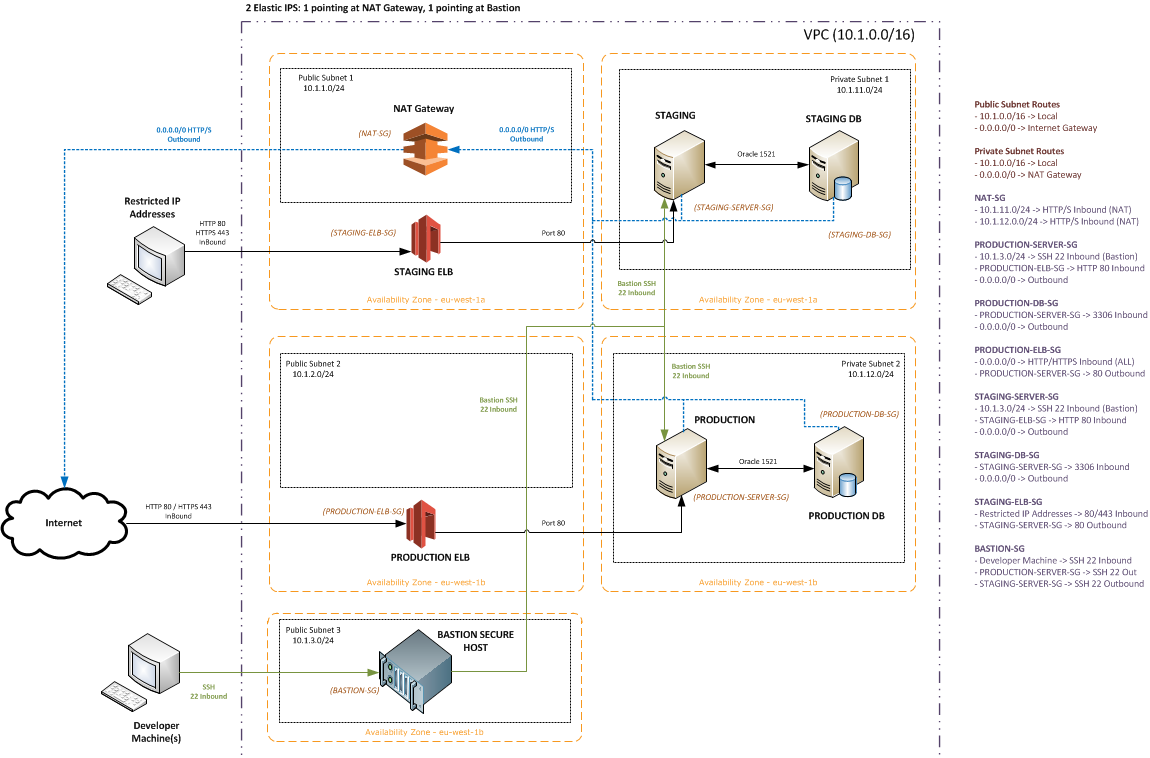
## [End of Solutions of Question 3]

## 

## QUESTION 4 (Solutions) [TOTAL MARKS: 25]

### Q 4(a) [14 Marks]

Essentially everything that was asked is just a replication of the following diagram:



### 

### Q 4(b) [4 Marks]

[Discussion of two of horizontal scaling, vertical scaling or elastic scaling from the course material]

### Q 4(c) [4 Marks]

**Internationalisation:**

Internationalisation is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes. This is typically the role of software developers to ensure that a software system is ready for localisation.

**Localisation:**

Localisation is the process of adapting internationalised software for a specific region or language by adding locale-specific components and translating text.

### Q 4(d) [3 Marks]

XHTML has a number of advantages over HTML. While it is less forgiving, it has a more rigid structure which still allows the developer to utilise the full functionality of HTML and CSS. By using XHTML we can ensure that the DOM model is preserved, which has a number of advantages for us if we are interacting with and modifying the DOM through JavaScript code or through Ajax. As we are also standards complaint using XHTML, we are more likely to have browser compatible code.

## [End of Solutions of Question 4]

## [END OF SOLUTIONS]