University of Glasgow

Thursday, 14 May 2015
2.00pm – 3.00pm
(Duration: 1 hour)

DEGREES of MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

WEB APPLICATION DEVELOPMENT 2

This examination paper is worth a total of 90 marks.

Each multiple choice question is worth 3 marks for correct answers. Incorrect answers will result in a 1 mark deduction.

Answer questions on the multiple choice sheet provided.

Answer all sections.

INSTRUCTIONS TO INVIGILATORS

Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.

At the end of the examination, return your answer sheet along with any script book used.

Section A (30 marks)

Q01. What sequence of commands would you issue to upload and share your code in a GitHub repository? You can assume the added code wont cause a conflict with existing code.

A. push commit add pull
B. push add commit pull
C. pull commit push add
D. pull add commit push

Q02. The typical architecture of a web application follows a:

A. Two tier architecture
B. Model View Controller
C. Three Tier architecture
D. Model View Template

Q03. What design pattern does the django web application use:

A. Factory Pattern
B. Model View Controller Pattern
C. Subject Observer Pattern
D. Model View Template Pattern

Q04. An overriding principle when design web applications is:

A. keep'em separated
B. concerns separation
C. separation of concerns
D. separation of content

Q05. The responsibilities of the middleware include: (select option where all correctly apply):

A. handle incoming requests from clients, request data from a database, return an html/xml response to the client
B. handle incoming request from clients, request data from a database, return a html/xml response to the server
C. handle incoming requests from servers, request data from the client, return an html/xml response the server
D. handle incoming requests from clients and servers, request data from server, return a html/xml response to the client and server.

Q06. What does HTTP stand for?

A. The good of all internet citizens
B. Hyper Text The Protocol
C. Hyper Text Transfer Protocol
D. Hyper Text Transfer Process

Q07. The two most frequently used in HTTP requests are:

A. PUSH and GET
B. GET and POST
C. PUSH and PULL
D. FETCH and POST

Q08. Which problem does HTTP suffer from:

A. entomophobia, fear of being crawled
B. statelessness, maintains no state
C. statefulness, maintains state
D. statuslessness, maintains no status

Q09. AJAX stands for:

A. Asynchronous JavaScript and eXtensible Markup Language
B. Amsterdam Football club
C. Asynchronous Java and Extensible Markup Language
D. Asynchronous JavaScript and eXtensible Makeup Language

Q10. AJAX enables us to (Select the option where all statements apply):

A. make requests without reloading the page, to receive and work with data from the server, using only xml data
B. make requests by reloading the page, to receive and work with data from the server, using any type data
C. make requests without reloading the page, to receive and work with data from the server, using any type data
D. make requests by reloading the page, to receive and work with data from the server, using only xml data

Section B (30 marks)

Q11. CSS stands for:

A. Cascading Style Sheet
B. Counterstrike Source Sheet
C. Cascading Styling Sheet
D. Cascade Style Sheet

Q12. Which selector applies to all tags with a class called blue and are paragraph elements:

A. .blue p
B. #blue p
C. blue p
D. $blue p

Q13. Which selector applies to all tags with the ID called blue and are div elements:

A. div blue
B. div #blue
C. blue div
D. $blue div

The following code applies to the next two questions (B4 and B5):

```
p {
      background-color: yellow;
      border: 1px solid black;
}


p .blue {
      background-color: blue;
}
```

Q14. Given the HTML code shown below how would the element be rendered:

```
<p id="blue">Hello World</p>
```

A. Hello world, with a solid border and a yellow background
B. Hello world, with no border and blue background
C. Hello world with a solid border and blue background
D. Hello world with no border and no background color

Q15. Given the HTML code shown below how would the element be rendered:

```
<p style="background-color: green">Goodbye World</p>
```

A. Goodbye world, with no border and green background
B. Goodbye world, with border and yellow background
C. Goodbye world with no border and no background color
D. Goodbye world with border and green background

Q16. By referencing the Javascript as follows in the HTML is an example of:

```
<head>
<link  type="text/javascript" href="broken.js">
</head>
```

A. inline referencing
B. embedded referencing
C. online referencing
D. external referencing

Q17. Assume that user has visited the site on previous occasions, and within the HTML on the site a piece of Javascript is referenced as follows:

```
<head>
<link  type="text/javascript" href="notbroken.js">
</head>
```

Select the option, where all apply:

A. page load times will be faster, content is separated from style, difficult to maintain
B. page load times will be slower, content is separated from style, easy to maintain
C. page load times will be faster, content is separated from style, easy to maintain
D. page load times will be slower, content is separated from style, difficult to maintain

Q18. Which line of JQuery code selects the element with a class brown and assigns an event to when the element is clicked:

```
A.  $(".brown").click( )
B.  $(".brown").onclick( )
C.  $("#brown").click( )
D.  $("#brown").onclick( )
```

Q19. Which line of code selects a paragraph element, and assigns a hover event to it, such that on hover the text turns blue, and off hover it turns red.

```
A. $("p").hover( function() { $(this).css('color', 'red’); },
        function() { $(this).css('color', 'blue'); };
B. $("p”).onhover( function() { $(this).css('color', 'red’); },
        function() { $(this).css('color', 'blue'); });
C. $("p").hover( function() { $(this).css('color', ‘blue’); },
        function() { $(this).css('color', ‘red'); });
D. $("p”).onhover( function() { $(this).css('color', ‘blue’); },
        function() { $(this).css('color', ‘red'); });
```

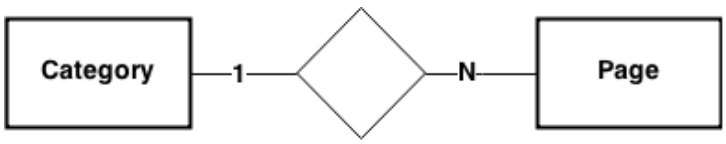
Q20. When writing JQuery code, it should be encapsulated by:

```
A. $(document).ready(function() {
        // JQuery code to be added in here.
   });
B. $().ready(function() {
        // JQuery code to be added in here.
   });
C. $(document).onready(function() {
        // JQuery code to be added in here.
   });
D. $().onready(function() {
        // JQuery code to be added in here.
   });
```

Section C (30 marks)

In this section all questions are related.

In `**app/models.py**', you need to create the models **Category** and **Page** based on the following ER diagram.



Each **Category** has a *name* of up to 128 characters, the number of *views* and *likes* are stored, along with a *slug line*. The name of each **Category** needs to be different, none can be the same.

The code for `**app/models.py**' is almost complete, but you need to complete it correctly, by selecting the correct lines of code.

```
from django.db import models
from django.template.defaultfilters import slugify
from django.contrib.auth.models import User

class Category(models.Model):
     <see Question C1>                                    (P1)
     views = models.IntegerField(default=0)
     likes = models.IntegerField(default=0)
     <see Question C2>                                    (P2)

     def save(self, *args, **kwargs):
          <see Question C3>                               (P3)
          super(Category, self).save(*args, **kwargs)

class Page(models.Model):
     <see Question C4>                                    (P4)
     title = models.CharField(max_length=128)
     url = models.URLField()
     views = models.IntegerField(default=0)

     def __unicode__(self):
          <see Question C5>                               (P5)
```

To finish off the implementations of the models select the correct lines of code.

Q21. What line of code should be inserted at P1:

A. name = models.CharField(max_length=128, unique=True)
B. name = models.CharField(unique=True)
C. name = models.CharField(default='', unique=True)
D. name = models.CharField(default='', max_length=128, unique=True)

Q22. What line of code should be inserted at P2:

A. slug = models.SlugField(default='',unique=True)
B. slug = models.SlugField(default='')
C. slug = models.SlugField()
D. slug = models.SlugField(blank=False)

Q23. What line of code should be inserted at P3:

A. slug = slugify(name)
B. slug = slugify(self.name)
C. self.slug = slugify(name)
D. self.slug = slugify(self.name)

Q24. What line of code should be inserted at P4:

A. category = models.CharField(default='',unique=True)
B. category = models.ForeignKey(Category)
C. category = models.ForeignKeyField(Category)
D. category = models.ManyToManyField(Category)

Q25. What line of code should be inserted at P5:

A. `return self.title`
B. `title`
C. `return title`
D. `self.title`

Given the models in `**app/models.py**', you now need to complete the view that shows all the pages associated with the category slug supplied, and list them in descending order. The template for the category page is at '**app/categories.html**' and the view is called, **category**.

```
<see Question C6>                                          (P6)
     context_dict = {}
     try:
          <see Question C7>                                (P7)
          <see Question C8>                                (P8)

          context_dict['category_name'] = category.name
          context_dict['pages'] = pages
          context_dict['category'] = category
     except Category.DoesNotExist:
          pass
     <see Question C9>                                     (P9)
```

Q26. What line of code should be inserted at P6:

A. `class category(request, category_name_slug):`
B. `def category(request):`
C. `def category(request, category_name_slug):`
D. `class category(request):`

Q27. What line of code should be inserted at P7:

A. category = Category.object.get(slug=category_name_slug)
B. category = Category.objects.filter(slug=category_name_slug)
C. category = Category.object.filter(slug=category_name_slug)
D. category = Category.objects.get(slug=category_name_slug)

Q28. What line of code should be inserted at P8:

A. pages = Page.objects.filter(category=category).order_by('views')
B. pages = Page.objects.filter(category=category).order_by('-views')
C. pages = Page.objects.get(category=category).order_by('views')
D. pages = Page.objects.filter(category=category).ordered_by(-'views')

Q29. What line of code should be inserted at P9:

A. return render(request, 'app/category.html', context_dict)
B. return render('app/categories.html', context_dict)
C. return render(request, 'app/categories.html', context_dict)
D. return render('app/category.html', context_dict)

Now add in `**app/urls.py**', you need to complete the url mappings for the category view.

```
from django.conf.urls import patterns, url
from rango import views

urlpatterns = patterns('',
      url(r'^$', views.index, name='index'),
      url(r'^about/$', views.about, name='about'),
      <see Question C10>                                    (P10)

      )
```

Q30. What line of code should be inserted at P10:

A. url(r'^category/(?P<category_name>[\w\-]+)/$', views.category, name='category'),
B. url(r'^category/(?P<category_name_slug>[\w\]+)/$', views.category, name='category'),
C. url(r'^category/(?P<category_name>[\w\]+)/$', views.category, name='category'),
D. url(r'^category/(?P<category_name_slug>[\w\-]+)/$', views.category, name='category'),