Grado en Ingeniero en Informática — Enero 2017
Programación Web

Examen de preguntas de respuesta corta

Responda de forma breve y razonada a las siguientes cuestiones:

1. Explique las diferencias fundamentales entre los métodos GET y POST del protocolo HTTP.

   RESPUESTA: El método GET permite pasar información del cliente al servidor en forma de texto que se añade a la dirección URL. El método POST envía información pero de forma codificada. GET debe usarse para enviar información reducida y sin problemas de seguridad, como por ejemplo un término a buscar. POST debe usarse para todo los demás. En particular no debe usarse GET para enviar información que pueda modificar la base de datos.

2. Explique qué hace el siguiente fragmento de código HTML:

```
<form action="http://www.google.com/search" method="get">
 <label>Google:<input type="search" name="q">
 <input type="submit" value="Search"> </label>
</form>
```

   RESPUESTA: Se trata de un formulario que obtiene del cliente un término y lo somete al buscador de Google.

3. Explique en qué consiste el *Document Object Model* (DOM) de un documento HTML.

   RESPUESTA: El DOM es una forma de representar en una estructura jerarquizada en forma de árbol el contenido completo de una página HTML de forma que pueda ser accedida desde un lenguaje de programación.

4. Suponga que tiene los siguientes modelos definidos en una aplicación Django denominada *polls*:

```
class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

   Usando la shell de Django ejecutamos el siguiente código:

```
>>> from polls.models import Question, Choice
>>> import datetime
>>> q = Question(question_text="Pregunta", pub_date=datetime.time.now())
>>> choice = Choice(choice_text="Respuesta 1", votes = 0, question = q)
>>> choice.save()
```

¿Es esta una manera correcta de proceder?

RESPUESTA: No. La relación definida en `question = models.ForeignKey(Question, on_delete=models.CAS` debe hacer referencia a un objeto de la base de datos y el objeto `q` acaba de ser creado y no está almacenado en la base de datos.

5. Suponga que tiene definidos los siguientes modelos en una aplicación Django denominada *blogs*:

```
class Author(models.Model):
    name = models.CharField(max_length=200)

class Entry(models.Model):
    headline = models.CharField(max_length=200)
    text = models.TextField()
    author = models.ManyToManyField(Author)
```

Suponga que tiene en la base de datos el autor de nombre "Autor 1" y que quiere añadir una nueva entrada de dicho autor. ¿Cómo lo realizaría usando la shell de Django?

RESPUESTA:

```
>>> from blogs.models import Entry, Author
>>> a = Author.objects.get(name = "Autor 1")
>>> entrada = Entry(headline = "Nueva entrada", text = "Texto ejemplo")
>>> entrada.save()
>>> entrada.author.add(a)
>>> entrada.save()
```

6. Considere los mismos modelos para la aplicación *blogs* de la pregunta anterior. ¿Cómo se podrían obtener todas las entradas realizadas por el autor "Autor 1"?

RESPUESTA: Existen diferentes formas. Por ejemplo:

```
>>> q = Entry.objects.filter(author__name="Autor 1")
```

7. Considere los mismos modelos para la aplicación *blogs* de la pregunta anterior. ¿Qué diferencias existen en la ejecución de las dos siguientes accesos a la base de datos?

Ejemplo 1:

```
>>> entry = Entry.objects.get(pk=1)
```

Ejemplo 2:

```
>>> entry = Entry.objects.filter(pk=1)
```

RESPUESTA: El caso 1 devolvería un objeto de tipo `Entry` en el caso de que exista un objeto con clave primaria 1. Si no existe tal objeto devolvería una excepción. En el caso 2 devolvería una lista con un única elemento de tipo `Entry` si existe el objeto con clave primaria 1 y en caso contrario devolvería una lista vacía.

8. Considere que tiene los definidos los siguientes modelos en una aplicación Django denominada *books*:

```
from django.db import models

class Publisher(models.Model):
    name = models.CharField(max_length=30)
    address = models.CharField(max_length=50)
    city = models.CharField(max_length=60)
```

Ponga un ejemplo de uso de la vista genérica basada en clase `ListView` para mostrar una lista de todos los objetos de la clase `Publisher`.

RESPUESTA:

```
# views.py
from django.views.generic import ListView
from books.models import Publisher

class PublisherList(ListView):
    model = Publisher
```

9. Explique qué función realiza el siguiente código en Javascript:

```
element = document.getElementById("element-id");
element.parentNode.removeChild(element);
```

RESPUESTA: Eliminaría el elemento de HTML cuyo id sea "element-id".

10. ¿Qué contenido mostraría un navegador después de cargar la siguiente página HTML?

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Page</h1>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

RESPUESTA: Mostraría:

```
My First Page
Hello World!
```

Tiempo de realización: **1** hora. Todas las cuestiones tienen la calificación de 1 punto.