Fundamentos de Computación

Ingeniería en Computación - Ingeniería en Informática
Departamento de Informática - UNSL - Año 2018

Práctico de Máquina

Procesamiento de códigos HTML

Fecha límite para la entrega del práctico: **1 de Junio de 2018**

El trabajo debe ser realizado en grupo de 2 alumnos.

Modalidad de trabajo:

La solución del práctico consiste en resolver la tarea asignada y entregar los códigos fuente creados junto con toda la documentación, información y archivos que considere necesarios a través del Aula Virtual.

La solución del práctico será evaluada con distintos archivos de pruebas y si la cátedra considera necesario, se realizará un coloquio del desarrollo del práctico. En caso de que la solución entregada presente errores, los mismos deberán ser corregidos y entregados en la instancia de recuperación.

1. Herramienta para detectar errores en códigos HTML

Una de las principales funciones de los navegadores web actuales es la de visualizar un archivo cuya extensión es HTML. HTML es un lenguaje de programación basado en etiquetas. Una etiqueta es una instrucción que define un bloque de código, de manera que el código delimitado por esa etiqueta cumple con ciertas propiedades.

Debido a que no es deseable ver errores de programación en una página web, los navegadores dejan de lado las fallas encontradas en el código, tratando de dar el mejor aspecto posible a la página. Por lo tanto, es útil tener una herramienta que detecte e informe los errores encontrados en el código HTML.

Se necesita desarrollar una aplicación en JAVA que dado un archivo HTML detecte errores en el código HTML de entrada. Para ello, la cátedra les provee un conjunto de archivos en JAVA que deberán ser utilizados para crear el detector de errores de HTML. A fin de entender e interpretar el código JAVA de los archivos entregados por la cátedra, en la Figura 1 se puede observar de modo esquemático la estructura de las clases contenida en los archivos.

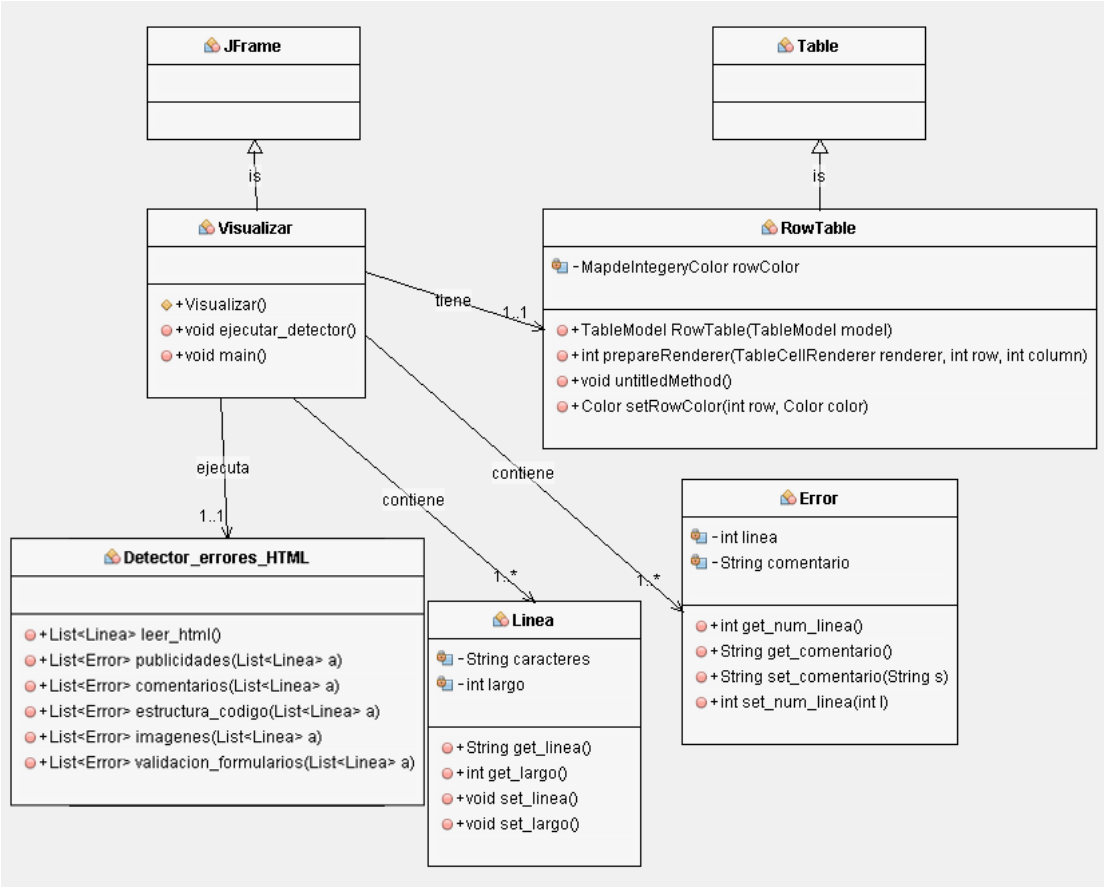

Figura 1: Diagrama de las clases JAVA provistas por la cátedra.

En los archivos entregados podrá encontrar:

- El método "leer_html" para leer línea a línea el archivo HTML y crear una lista en JAVA con cada línea ("$List < Linea >$").
- La clase "Visualizar" para la visualización por pantalla del código html y de los errores.

Cada grupo deberá implementar un método de la clase **Detector_errores_HTML** para detectar un tipo de error específico en el código HTML. El método será asignado a cada grupo por sorteo en una de las clases prácticas de la materia y su explicación se puede consultar en el punto 2 de este práctico.

Al detectar un error, se debe informar con un mensaje el tipo de error y donde se produjo ($Linea$) el error dentro del código HTML.

Aclaraciones

- Los mensajes de los errores deben dar la mayor información posible al usuario con el objetivo de ayudarlo en la corrección de los mismos.

- A fin de obtener una mayor legibilidad en la implementación de la aplicación, cuando agregue código JAVA a los archivos entregados por la cátedra, debe incluir comentarios en el código además de información interpretables por la herramienta JAVADOC.
- No se puede utilizar ningún tipo de librería especial para el control del código HTML.

2. Errores a detectar:

2.1. Bloqueador de publicidades o sitios engañosos

Actualmente los navegadores traen complementos para evitar la visualización de anuncios no deseables en las páginas. Estos complementos funcionan de la siguiente manera: al cargar una página si se detecta la referencia a un contenido marcado por un filtro (listado cuyo contenido son patrones de anuncios), se borra la capa (`div`) que contiene el enlace y que es a su vez es la más cercana.

Por ejemplo, para el siguiente código HTML:

```
<div id="pub" style="color: white;" align="center">
    <b>Publicidad se cerrara en
    <span>4</span>
    segundos.</b>
      
    <a href="https://www.blogger.com/adsdiv"
    onclick="closeMyAd()">
    </a>
</div>
```

Se procede a borrar todo el contenido de la etiqueta `div` con identificador `pub` debido a que **adsdiv** pertenece al filtro.

Dada una lista reducida de patrones considerados como portadores de anuncios (Archivo: *EasyList*); detectar la línea en la página que hace referencia a ellas y si se trata de un bloque detectar el inicio y el fin del mismo.

2.2. Error dentro de un comentario

Se necesita detectar las situaciones en las cuales una etiqueta sigue abierta debido a que alguna etiqueta del bloque se encuentra dentro de un comentario.

Código incorrecto:	Forma correcta:
Error 1 <!-- Comentario descuidado 1 --> - Manzanas - Kiwi - Papas **Error 2** - Manzanas - Kiwi - Papas <!-- Comentario descuidado 2 -->	<!-- Comentario... --> - Manzanas - Kiwi - Papas <!-- Comentario... -->

Algunas etiquetas HTML no necesitan la regla de anidamiento, y por lo tanto, no hace falta una etiqueta que cierre el bloque determinado por ella (< /___ >). Las etiquetas que consideraremos como exclusión de esta norma son:

- El salto de línea (
).
- La inserción de una imagen (<img>).
- La inserción de una animación flash (<embed>).
- La creación de un cuadro (<frame>).
- Una entrada de un formulario (<input>).
- El campo Keygen de un formulario (<keygen>).

Las etiquetas válidas en HTML están listadas en el anexo.

2.3. Estructura de HTML5 y de la etiqueta HEAD

La estructura de un archivo HTML5 puede ser expresada en BNFE de la siguiente manera:

```
<html5>:: = <!doctype html>
            <html>
            <head>
            <body>
```

```
                <header>
                {<section>}
                [<aside>]
                <footer>
        </body>
        </html>
```

Además, una etiqueta `<head>` se puede expresar en el BNFE de la siguiente forma:

```
<head> :: = <head> <title>
                [<base>]
                [<isindex>]
                {{<script>}{<style>}{<meta>}{<link>}{<object>}}
            </head>
```

Se pide:

- Detectar si el archivo HTML respeta el estándar para un archivo `html5` y para el formato del `head`.
- Comprobar la existencia del atributo: `<meta charset="utf-8">` en la etiqueta `head`.
- Si el atributo `rel = "stylesheet"` en `<link>` está definido, comprobar que su contenido tenga un archivo con extensión "css", además verificar que no se encuentre repetido.
- Si existe el atributo `src` en la etiqueta `<script>` comprobar que el contenido tenga un archivo con extensión "js", además verificar que no se encuentre repetido.

Nota: En el código BNFE, la negrita es usada para los símbolos terminales.

2.4. Imágenes

La etiqueta (`img`) sólo pueden estar dentro del entorno definido por alguna de las siguientes etiquetas: `header`, `section`, `aside` o `footer`.

Se pide controlar dentro la etiqueta `img`:

A La propiedad anterior.

B La existencia del atributo `src`. Además, se debe comprobar que el contenido del atributo `src` sea una dirección de internet activa. Si la dirección es relativa debe hacer uso de la etiqueta `base`. Ejemplo:

```
<base href="https://www.geeksforgeeks.org/images">
```

C El archivo referenciado por el atributo `src` debe tener una de las siguientes extensiones: jpg, jpeg, png, gif, svg.

D Si está definido el atributo `alt`, verificar que no sea vacío.

Nota: Hacer uso de las expresiones regulares de Java para los puntos C y D.

2.5. Validación de formularios

Este apartado detecta errores y agrega información al archivo de entrada HTML.

La etiqueta (`input`) especifica un campo de entrada donde el usuario puede ingresar datos dentro de un formulario. Una estructura básica de esta etiqueta sería la siguiente:

```
<input type="__" name="__" id="_"

pattern="<Expresión_regular>"/>
```

Se pide:

- Crear un archivo HTML utilizando el archivo de entrada e incorporando cada una de las `<Expresión_regular>` para los siguientes valores del atributo `name`: Nombre, Apellido, DNI, CUIL, Correo electrónico, Teléfono, Fecha de nacimiento,Comentarios.
- Además de agregar la expresión regular se debe incluir un comentario indicando el formato correcto para cada campo del formulario.
- Verificar que los valores de los atributos:
  - No sean vacíos.
  - No contengan ningún espacio en blanco y ninguno de los siguientes símbolos: ”, ’, =, ¿, ¡, ‘.
  - Estén encerrados por comillas.

3. Anexo: Listado de etiquetas válidas en HTML5

Tags:	Tags:	Tags:	Tags:
!–	dialog	label	samp
!doctype	div	legend	script
a	dl	li	section
abbr	dt	link	select
address	em	main	small
area	embed	map	source
article	fieldset	mark	span
aside	figcaption	menu	strong
audio	figure	menuitem	style
b	footer	meta	sub
base	form	meter	summary
bdi	h1	nav	sup
bdo	h2	noscript	table
blockquote	h3	object	tbody
body	h4	ol	td
br	h5	optgroup	textarea
button	h6	option	tfoot
canvas	head	output	th
caption	header	p	thead
cite	hr	param	time
code	html	picture	title
col	i	pre	tr
colgroup	iframe	progress	track
datalist	img	q	u
dd	input	rp	ul
del	ins	rt	var
details	kbd	ruby	video
dfn	keygen	s	wbr