Grado en Ingeniería Informática del Software

Software y estándares para la Web

P5. XML : ESQUEMAS Y PROCESAMIENTO

Escuela de Ingeniería Informática
Universidad de Oviedo

Contenido

Objetivos 2

Temática del proyecto: MotoGP Desktop 2

Ejercicio 1: Uso de XML Esquemas para validar los archivos XML del circuito 3

Tarea 1.1: Generación del archivo circuito.xsd a partir del archivo circuito.dtd 3

Guía de la solución de la tarea 1.1 3

Tarea 1.2: Modificar el archivo circuito.xsd para especificar archivos XML usando las características de los XML esquemas 3

Guía de la solución de la tarea 1.2 3

Tarea 1.3: Crear un archivo circuitoEsquema.xml a partir del archivo circuito.xml 4

Guía de la solución de la tarea 1.3 4

Tarea 1.4: Validar el archivo circuitoEsquema.xml con circuito.xsd 4

Guía de la solución de la tarea 1.4 4

Tarea 1.5: Generar gráficamente el árbol DOM del archivo circuito.xsd denominado “esquemaCircuito.svg” 4

Guía de la solución de la tarea 1.5 4

Ejercicio 2: Generar archivos KML a partir del archivo circuitoEsquema.xml usando el lenguaje Python 4

Guía de la solución del ejercicio 2 5

Ejercicio 3: Generar el archivo de altimetría del circuito en formato SVG a partir del archivo circuitoEsquema.xml usando el lenguaje Python 6

Guía de la solución del ejercicio 3 6

Ejercicio 4: Generar el archivo InfoCircuito.html a partir del archivo circuitoEsquema.xml usando el lenguaje Python 7

Guía de la solución del ejercicio 4 8

Resultados 8

Recuerda 9

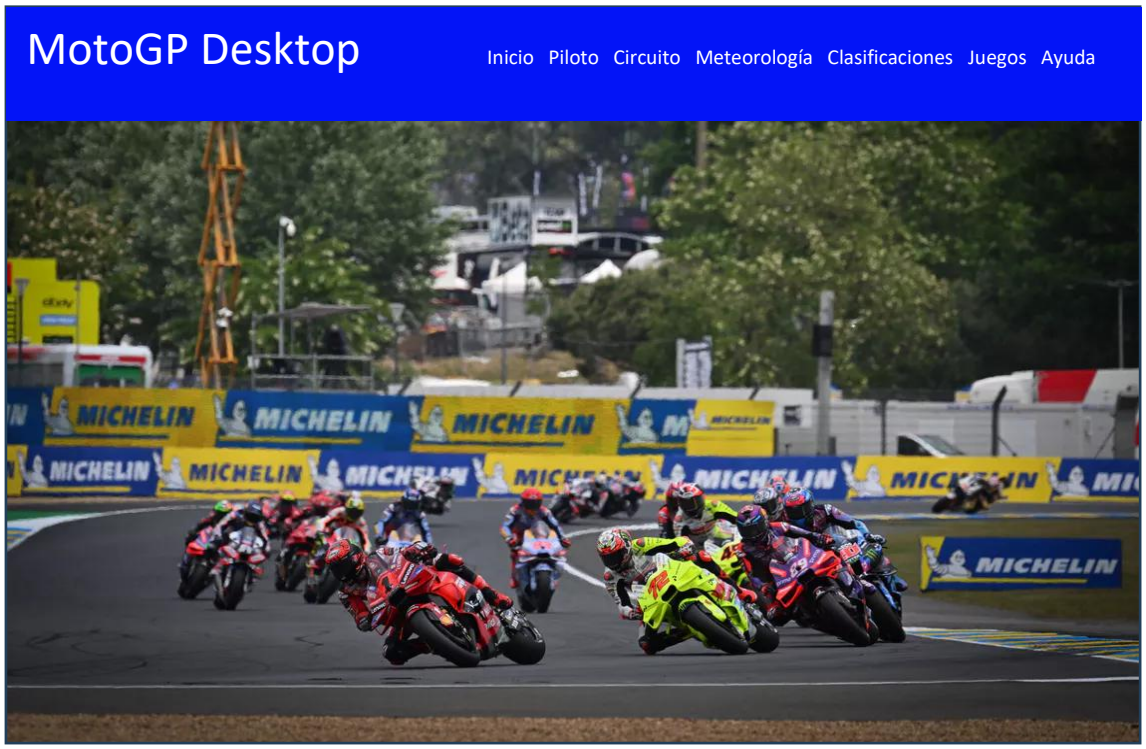
Objetivos

En esta práctica se va a realizar:

- El uso de los estándares KML y SVG (Teoría “Introducción a XML”)
- La validación del documento XML con un XML Schema (Teoría “XML Schemas”)
- El lenguaje XPath (Teoría “XPath”)
- El procesamiento del documento XML y la generación de nuevos documentos XML (Teoría “Procesamiento y generación de XML”)

Temática del proyecto: MotoGP Desktop

Se va a crear MotoGP-Desktop para alojar el proyecto de la asignatura. El proyecto es evolutivo y será creado, completado y modificado en las diferentes prácticas de la asignatura.



Ejercicio 1: Uso de XML Esquemas para validar los archivos XML del circuito

Se tienen que usar XML esquemas para validar los archivos del circuito

Tarea 1.1: Generación del archivo circuito.xsd a partir del archivo circuito.dtd

Utilizando una herramienta se genera una primera versión del archivo XML esquema denominado circuito.xsd a partir del archivo circuito.dtd

Guía de la solución de la tarea 1.1

Se aconseja utilizar como herramienta "***Visual Studio 2022 Community Edition***", explicado en el tema de teoría "XML Schemas".

https://visualstudio.microsoft.com/es/vs/community/

Se abre el archivo **circuito.dtd** con la herramienta, aparece un **menú denominado XML** y se elige la opción "*Esquemas …*" para genera el archivo **circuito.xsd**

El problema es que el archivo generado **circuito.xsd** tiene las mismas limitaciones de los DTD y ninguna de las ventajas de los XML esquema.

Tarea 1.2: Modificar el archivo circuito.xsd para especificar archivos XML usando las características de los XML esquemas

Se solicita modificar el archivo **circuito.xsd** usando las características de los XML esquemas. Se deben utilizar tipos de datos, rangos y restricciones. Se generará un archivo denominado **informeXMLesquema.pdf** donde se deben explicar los cambios realizados.

Guía de la solución de la tarea 1.2

Se debe editar el archivo **circuito.xsd** obtenido en la tarea anterior y modificarlo para que use las características propias de los XML esquema.

Los DTD solamente tienen un tipo de datos "***string***" denominado PCDATA (en elementos) y CDATA (en atributos). Sin embargo, XML esquema tiene una amplia variedad de tipos de datos que permiten especificar mucho mejor los archivos XML.

Deben usarse tipos de datos para especificar valores enteros, float, fecha, hora, duración, etc…

Los DTD solamente especifican rangos con "?", "+" , "*". Sin embargo, XML esquema permite también especificar rangos entre dos valores numéricos. También se deben poner restricciones al rango que pueden tomar los valores.

La herramienta también nos puede cambiar la codificación, asegurar que se sigue en "UTF-8".

Los elementos deben tener el prefijo "**http://www.uniovi.es"**

Tarea 1.3: Crear un archivo circuitoEsquema.xml a partir del archivo circuito.xml

Se crea un archivo **circuitoEsquema.xml** que contiene la información del circuito, pero con un enlace al archivo XML esquema denominado **circuito.xsd**, utilizando como prefijo de los elementos ***"http://www.uniovi.es"***

Guía de la solución de la tarea 1.3

El archivo **circuitoEsquema.xml** es el archivo **circuito.xml** al que se ha eliminado la línea que contiene el enlace al archivo **circuito.dtd** y debe modificarse por el enlace al archivo **circuito.xsd**

Consultar los ejercicios resueltos del tema de teoría "XML Schemas".

Tarea 1.4: Validar el archivo circuitoEsquema.xml con circuito.xsd

Se debe validar el archivo "circuitoEsquema.xml" con el archivo XML esquema "circuito.xsd"

Guía de la solución de la tarea 1.4

Utilizar una herramienta o un editor con validación de XML esquemas para comprobar que el archivo circuitoEsquema.xml es válido con circuito.xsd

Se aconseja usar "Visual Studio Code" con el plug-in "XML language tools red hat".

Tarea 1.5: Generar gráficamente el árbol DOM del archivo circuito.xsd denominado "esquemaCircuito.svg"

El archivo XML esquema denominado **circuito.xsd** es un archivo XML y por tanto se puede visualizar su árbol DOM con un archivo denominado "**esquemaCircuito.svg**" generado con la herramienta "xml2svg.exe"

Se debe generar el archivo "esquemaCircuito.svg", también se debe generar el archivo "**esquemaCircuito.pdf**"

Guía de la solución de la tarea 1.5

Usar la herramienta "xml2svg.exe" para generar el archivo "esquemaCircuito.svg" desde una terminal en línea de comandos. Usar el navegador Opera para generar el PDF.

Ejercicio 2: Generar archivos KML a partir del archivo circuitoEsquema.xml usando el lenguaje Python

Se deben generar un archivo en formato KML denominados "**circuito.kml**" a partir del archivo "**circuitoEsquema.xml**". Debe usarse obligatoriamente el lenguaje **Python**, se debe crear un archivo denominado "**xml2kml.py**".

Los archivos KML representan la planimetría del circuito y pueden visualizarse con distintas herramientas, se debe capturar la imagen de la planimetría del circuito en un archivo en formato

PDF denominados "**planimetria.pdf**". El archivo KML se mostrará sobre la imagen fotográfica del circuito.

Guía de la solución del ejercicio 2

El formato de los archivos KML se explica en el *tema 1: Introducción a XML*. Este formato puede representar distintos elementos gráficos.

Cómo ejemplo de representación de un punto puede verse el ejemplo **Oviedo.kml** en el *tema 1: Introducción a XML*.

También se pueden representar líneas, puede verse en el ejemplo **N1612080.kml** en el *tema 6: Procesamiento y generación de XML* (06-Ejemplos\Nikon).

Además, puede verse una combinación de puntos y líneas en el ejemplo **rutaOviedo.kml** en el *tema 6: Procesamiento y generación de XML* (06-Ejemplos\Python).

Para el desarrollo en el lenguaje Python debe importarse la biblioteca:

```
import xml.etree.ElementTree as ET
```

esta biblioteca permite acceder al árbol DOM de los archivos XML y obtener las coordenadas del circuito.

Para el manejo de esta biblioteca para leer el archivo "**circuitoEsquema.xml**" y construir el árbol DOM en memoria, se deben consultar los ejercicios:

a) Tema de teoría "*XPath*", ejercicio "**02010-XPath.py**". Ilustra el manejo de expresiones XPath, que **es obligatorio su uso para obtener los elementos y atributos del árbol DOM**.

b) Tema de teoría "*Procesamiento y generación de XML*", ejercicio "**Python/02000-XML.py**". Ilustra el recorrido de un árbol DOM completo, tanto de de archivos XML con prefijo y sin prefijo.

Para generar el código del archivo "**circuito.kml**" se puede hacer de **dos formas**;

1. Tema de teoría "*Procesamiento y generación de XML*", ejercicio "**Nikon/Nikon-NMEA-KML.py"**

   En este ejemplo se generan los archivos KML utilizando "generación de código basado en plantillas". El programa se debe realizar siguiendo la siguiente organización:

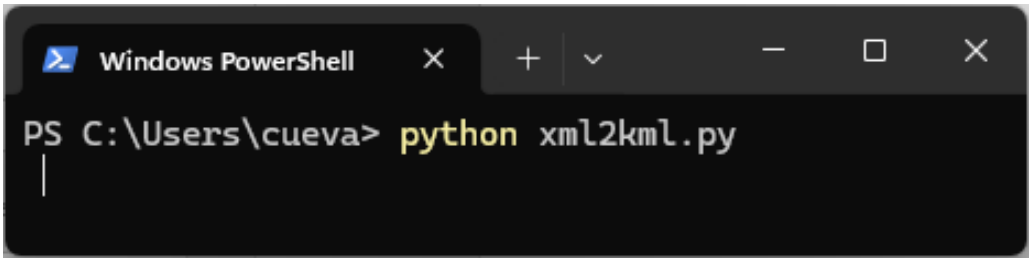
   - Leer el archivo "**circuitoEsquema.xml**" y generar en memoria el árbol DOM
   - Escribir el archivo **"circuito.kml**"
     - Escritura del **prólogo** con las etiquetas de encabezado del archivo KML
     - Escritura de las coordenadas del circuito. Para extraerlas del árbol DOM se deben utilizar obligatoriamente **expresiones xPath**.
     - Escribir el **epílogo** con las etiquetas de cierre del archivo KML

2. Tema de teoría "*Procesamiento y generación de XML*", ejercicio "**Python/02020-KML.py**", en este ejemplo se usa una clase para generar archivos KML.

   - Leer el archivo "**circuitoEsquema.xml**" y generar en memoria el árbol DOM

- Escribir el archivo **"circuito.kml"**
  - Usar la clase Kml para generar las etiquetas de encabezado del archivo KML
  - Usar los métodos de la clase Kml para la escritura de las coordenadas del circuito. Para extraerlas del árbol DOM se deben utilizar obligatoriamente **expresiones xPath**.
  - Usar los métodos de la clase Kml para las etiquetas de cierre del archivo KML

El programa "**xml2kml.py**" se ejecutará en línea de comandos en una "ventana cmd" o "Terminal " usando el intérprete del lenguaje Python. Puede hacerse en los sistemas operativos Windows, Linux o MacOS.



Puede utilizarse "*Google Earth*" con la opción "**Archivo-> Abrir archivo KML local**" para visualizar el archivo KML con la imagen del circuito. Si se utiliza el navegador Opera se puede capturar la pantalla en PDF. De esta forma se comprueba que el archivo KML generado se superpone correctamente con el circuito.

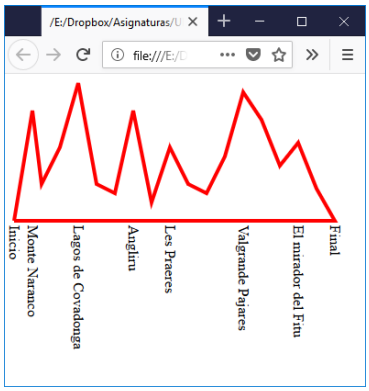
Ejercicio 3: Generar el archivo de altimetría del circuito en formato SVG a partir del archivo circuitoEsquema.xml usando el lenguaje Python

Se debe generar un archivo en formato SVG denominados "**altimetria.svg**" a partir del archivo "**circuitoEsquema.xml**". Debe usarse obligatoriamente el lenguaje Python. Se debe crear un archivo denominado "**xml2altimetria.py**"

El archivo SVG representan un perfil con la altimetría del circuito y pueden visualizarse con distintos agentes de usuario (navegadores). Se debe capturar la imagen de la altimetría del circuito en un archivo en formato PDF denominado **altimetria.pdf**.

Guía de la solución del ejercicio 3

El formato de los archivos SVG se explica en el *tema 1: Introducción a XML*. Un perfil altimétrico se puede construir con el elemento gráfico poli-línea **("polyline")** , textos verticales y horizontales, así como líneas horizontales y verticales para mostrar la escala. A continuación, ejemplo de poli-línea con texto vertical ("*tema 1: Introducción a XML*", archivo ***polilinea-texto.svg***).



Para el desarrollo en el lenguaje Python se debe importar la biblioteca:

```
import xml.etree.ElementTree as ET
```

esta biblioteca permite acceder al árbol DOM de los archivos XML y obtener las distancias y altimetrías del circuito por medio de expresiones xPath (obligatorio).

En el tema de teoría "*Procesamiento y generación de XML*", ejercicio "**Python/02030-SVG.py**", se usa una clase para generar archivos SVG. El procedimiento es el siguiente:

- Leer el archivo "**circuitoEsquema.xml**" y generar en memoria el árbol DOM
- Escribir el archivo **"altimetria.svg**"
    - Usar la clase Svg para generar las etiquetas de encabezamiento del archivo SVG
    - Usar los métodos de la clase Svg para la escritura de los datos de distancia y altitud de los puntos del circuito. Para extraerlos del árbol DOM se deben utilizar obligatoriamente **expresiones xPath**.
    - Usar los métodos de la clase Svg para las etiquetas de cierre del archivo SVG

Para crear el archivo SVG se puede utilizar una polilínea siendo sus vértices los puntos del circuito. Se puede cerrar la polilínea para hacer un efecto suelo en el perfil. También se puede rellenar.

Para convertir los archivos SVG a PDF se puede usar el navegador Opera y crear el archivo "**altimetría.pdf**".

Ejercicio 4: Generar el archivo InfoCircuito.html a partir del archivo circuitoEsquema.xml usando el lenguaje Python

Se debe generar un archivo en formato HTML denominados "**InfoCircuito.html**" a partir del archivo "**circuitoEsquema.xml**". El archivo HTML deberá contener toda la información del archivo XML. El archivo HTML deberá enlazar al archivo CSS "**estilo.css**" del proyecto "MotoGP Desktop". Se deben cumplir los estándares de HTML y CSS. El archivo HTML debe ser adaptable y accesible.

Debe usarse obligatoriamente el lenguaje Python. Se debe crear un archivo denominado "**xml2html.py**", usando una clase denominada Html para generar el código HTML. La extracción

de la información del árbol DOM del archivo XML debe hacerse obligatoriamente usando expresiones XPath.

No se incorporará al archivo HTML la información del punto de origen y los tramos del circuito.

El archivo "**InfoCircuito.html**" será utilizado en las próximas sesiones de prácticas para incorporar información en el documento "**circuito.html**" del proyecto MotoGP-Desktop.

Guía de la solución del ejercicio 4

Para el desarrollo en el lenguaje Python se debe importar la biblioteca:

```
import xml.etree.ElementTree as ET
```

esta biblioteca permite acceder al árbol DOM de los archivos XML y obtener toda la información del circuito por medio de expresiones xPath (obligatorio).

Se debe crear una clase Html de forma similar a las clases Kml y Svg de los ejercicios anteriores. Esta clase se utilizará para generar código.

Resultados

En la carpeta XML se deben añadir los siguientes archivos a los que ya se tienen de la práctica 4:

- circuitoEsquema.xml
- circuito.xsd
- informeXMLesquema.pdf
- esquemaCircuito.svg
- esquemaCircuito.pdf

- circuito.kml
- xml2kml.py
- planimetria.pdf

- altimetria.svg
- xml2altimetria.py
- altimetria.pdf

- InfoCircuito.html
- xml2htmll.py

Recuerda

Se requiere el uso correcto de los elementos HTML5 y de las propiedades de los módulos CSS.

Las siguientes condiciones son de obligado cumplimiento en todo el proyecto:

- **TODOS** los documentos html que componen el proyecto deben ser HTML5 válidos y sin advertencias utilizando el validador de lenguajes de marcado del W3C
- Se deben utilizar las etiquetas semánticas de HTML5 (section, article, etc.) y no está permitido el uso de bloques anónimos (**div**). En aquellos ejercicios en los que se permita el uso de bloques anónimos (**div**) estará indicado en el enunciado del ejercicio; **fuera de esos ejercicios, el uso de bloques anónimos no está permitido.**
- Se deben utilizar selectores de CSS específicos, el uso de selectores id y class no está permitido. **En aquellos ejercicios en los que se permita el uso de los selectores class o id estará indicado expresamente en el enunciado del ejercicio.**
- **TODAS** las reglas de todas las hojas de estilo deben estar precedidas por un comentario donde se indique la especificidad del (o los) selector(es) de la regla.
- **TODAS** las hojas de estilo que se utilizan en el sitio web deben ser validas utilizando el validador CSS del W3C
- **TODAS** las hojas de estilo deben tener 0 advertencias.
    - Recuerda seleccionar en “Más opciones “el informe de “Todas las advertencias” dentro de las opciones del validador CSS del W3C.
    - Excepcionalmente se permite las advertencias referidas a la verificación de los colores (color y background-color). **OBLIGATORIAMENTE** se debe indicar mediante un comentario en la regla de la hoja de estilo afectada la herencia de colores garantizando que la advertencia ha sido comprobada, verificada y garantizando que no provoca efectos laterales no deseados.
    - Excepcionalmente se permiten las advertencias referidas a la redefinición de propiedades derivadas del uso de @media-queries. **OBLIGATORIAMENTE** se debe indicar mediante un comentario en las reglas de la hoja de estilo afectada que propiedades se están redefiniendo.
- Se debe garantizar la adaptabilidad y realizar su verificación para todos los documentos que componen el proyecto.
    - Se deben utilizar medidas relativas en las hojas de estilo.
- Se debe garantizar la accesibilidad del proyecto mediante los test de las herramientas de accesibilidad para el nivel AAA de las WCAG 2.0 con 0 errores de modo automático u 0 errores de contraste, en todos los documentos que lo componen.

El no cumplimiento de las características anteriores derivará en la invalidación del proyecto.