

## 4.1. Representación de datos XML (CSS, XSL)



# Índice

---

Objetivos .....	3
Documentos XML .....	4
Formato de los documentos XML .....	4
¿Cómo se ven en el navegador?.....	6
Aplicar estilo al documento XML .....	11
Aplicar hojas de estilo CSS .....	11
Trasformaciones XSL .....	13
Usar XML, XSL y CSS.....	21
Filtrar los resultados con XPath.....	22
Despedida .....	27
Resumen.....	27

# Objetivos

En esta lección perseguimos los siguientes objetivos:

- Conocer la estructura de un documento XML.
- Construir documentos XML.
- Añadir estilo estético a la salida de los documentos XML, utilizando hojas de estilo CSS.
- Añadir estilo estético a la salida de los documentos XML, utilizando transformaciones XSL (*Extensible Stylesheet Language*).
- Realizar filtros utilizando XPath.

# Documentos XML

## Formato de los documentos XML

Los **ficheros XML** (*Extended Markup Language*) proveen un formato ligero para el intercambio de datos, lo que resulta especialmente interesante en aplicaciones web.

Los documentos SGML siguen un esquema jerárquico, compuesto por etiquetas con apertura y cierre que tienen la siguiente estructura: `<etiqueta> contenido </etiqueta>`. Cada etiqueta puede portar otras subetiquetas, formando así la jerarquía.

HTML y XML se basan en el estándar SGML. Tienen su origen en el lenguaje SGML (*Standard Generalised Mark-up Language*) definido por la norma ISO 8879 en 1986. Se trata de un estándar para lenguajes de marcado, con el objetivo de definir documentos independientes de las plataformas hardware y software.

**Una etiqueta en un documento XML y representa un nodo o elemento que puede estar compuesto por otros nodos.**

En el siguiente ejemplo tenemos un documento que representa los datos de varios cruceros.

```
<?xml version="1.0" encoding="UTF-8"?>
<cruceros>
  <crucero codigo="CRUMED21">
    <destino>Mediterraneo (Grecia, Italia)</destino>
    <detalles>
      <cia>Costa cruceros</cia>
      <dias>6 días</dias>
      <fechaSalida>2018-12-26</fechaSalida>
    </detalles>
    <escalas>
      <escala dia="1">
        <parada>Venecia</parada>
        <llegada></llegada>
        <salida>18:00</salida>
      </escala>
      <escala dia="2">
        <parada>Navegación</parada>
        <llegada></llegada>
        <salida></salida>
      </escala>
      <escala dia="3">
        <parada>Agostini</parada>
        <llegada>7:00</llegada>
        <salida>14:00</salida>
      </escala>
      <escala dia="4">
        <parada>Santorini</parada>
        <llegada>9:00</llegada>
        <salida>20:00</salida>
      </escala>
      <escala dia="5">
```

```

        <parada>Bari</parada>
        <llegada>8:00</llegada>
        <salida>14:00</salida>
    </escala>
    <escala dia="6">
        <parada>Venecia</parada>
        <llegada>8:30</llegada>
        <salida></salida>
    </escala>
</escalas>
</crucero>
<crucero codigo="CRUATL22">
    <destino>Atlántico (España, Marruecos, Portugal)</destino>
    <detalles>
        <cia>MSC Cruceros</cia>
        <dias>7 días</dias>
        <fechaSalida>2019-02-13</fechaSalida>
    </detalles>
    <escalas>
        <escala dia="1">
            <parada>Barcelona</parada>
            <llegada></llegada>
            <salida>18:00</salida>
        </escala>
        <escala dia="2">
            <parada>Navegación</parada>
            <llegada></llegada>
            <salida></salida>
        </escala>
        <escala dia="3">
            <parada>Casablanca</parada>
            <llegada>7:00</llegada>
            <salida>22:00</salida>
        </escala>
        <escala dia="4">
            <parada>Navegación</parada>
            <llegada></llegada>
            <salida></salida>
        </escala>
        <escala dia="5">
            <parada>Santa Cruz de Tenerife</parada>
            <llegada>9:00</llegada>
            <salida>16:00</salida>
        </escala>
        <escala dia="6">
            <parada>Funchal</parada>
            <llegada>8:00</llegada>
            <salida>19:00</salida>
        </escala>
        <escala dia="7">
            <parada>Barcelona</parada>
            <llegada>17:00</llegada>
            <salida></salida>
        </escala>
    </escalas>
</crucero>
</cruceros>

```

La primera línea del documento (cabecera), es la que identifica el documento como tipo XML y donde se especifica la versión, tipo de codificación, etc. Luego va la apertura del nodo raíz que encierra el resto de los nodos.

```
<?xml version="1.0" encoding="UTF-8"?>
<cruceros>
    .....
</cruceros>
```

El documento del ejemplo está concebido para portar la información de varios cruceros, por lo que el nodo raíz `<cruceros>` contendrá un conjunto de nodos `<crucero>`. Ésta podría ser la estructura para una base de datos documental, donde cada etiqueta `<crucero>` encierra la información necesaria para la redacción de un documento que describe un crucero.

Los documentos XML, además de utilizarse para portar datos, sirven para establecer valores de configuración en las aplicaciones web. Un ejemplo es el archivo *persistence.xml* cuya función vimos anteriormente.

## ¿Cómo se ven en el navegador?

Los documentos XML están concebidos para portar datos, pero no la presentación o el diseño de los mismos.

En este apartado vamos a ver cómo se muestran los documentos XML en los principales navegadores.

Te proponemos que crees el documento XML dentro de un proyecto Java. Aunque no vamos a programar en Java, sí lo haremos en la siguiente lección, de modo que ya tendremos el proyecto preparado.

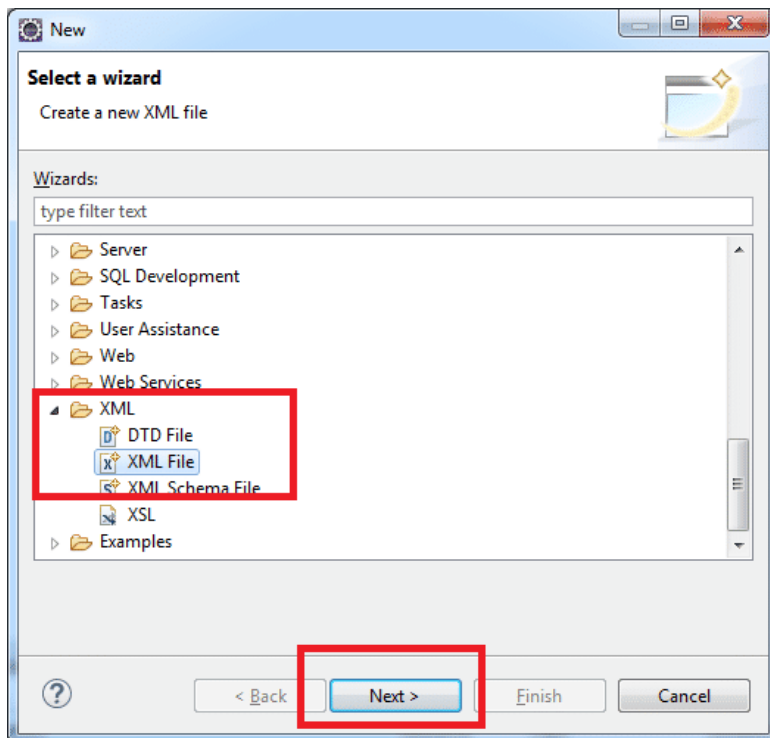
Sigue estos pasos:

### 1. Crea un proyecto Java estándar:

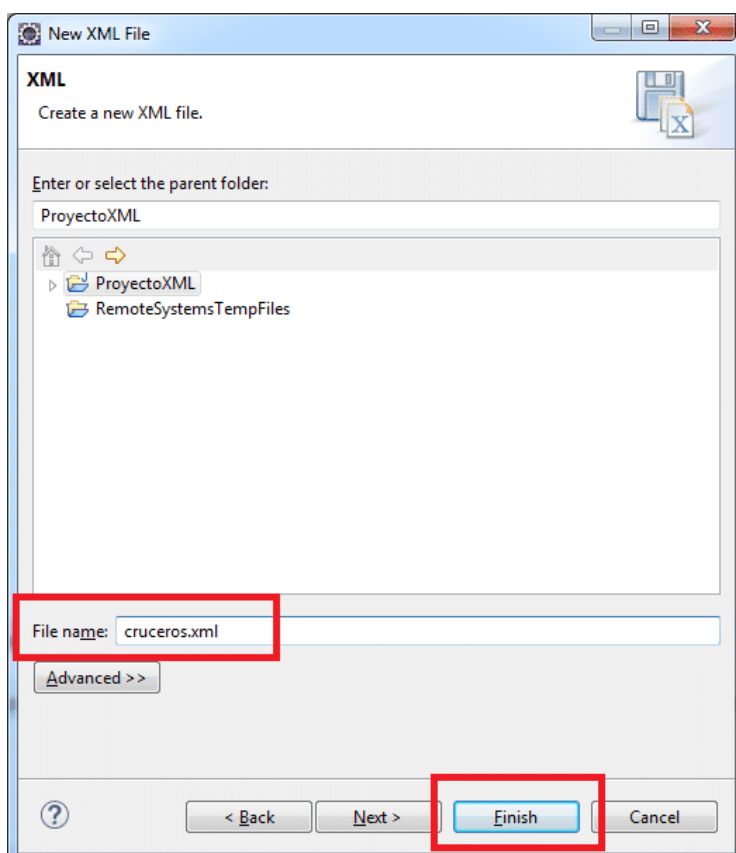
- Cambia a la perspectiva Java.
- Selecciona en el menú *File / New / Java project*.
- Escribe *ProyectoXML* como nombre del proyecto y pulsa *Finish*.

### 2. Crea un nuevo archivo XML dentro del proyecto:

- Haz clic derecho sobre el nombre del proyecto y selecciona *New / Other* en el menú contextual.
- Ahora te encuentras en el cuadro de diálogo *New*. Despliega la carpeta *XML*, escoge la opción *XML File* y pulsa *Next*.



- Escribe *cruceros.xml* como nombre del archivo y pulsa *Finish*.





**3. Copia y pega el contenido del archivo XML desde aquí:**

```

<?xml version="1.0" encoding="UTF-8"?>
<cruceros>
  <crucero codigo="CRUMED21">
    <destino>Mediterraneo (Grecia, Italia)</destino>
    <detalles>
      <cia>Costa cruceros</cia>
      <dias>6 días</dias>
      <fechaSalida>2018-12-26</fechaSalida>
    </detalles>
    <escalas>
      <escala dia="1">
        <parada>Venecia</parada>
        <llegada></llegada>
        <salida>18:00</salida>
      </escala>
      <escala dia="2">
        <parada>Navegación</parada>
        <llegada></llegada>
        <salida></salida>
      </escala>
      <escala dia="3">
        <parada>Agostini</parada>
        <llegada>7:00</llegada>
        <salida>14:00</salida>
      </escala>
      <escala dia="4">
        <parada>Santorini</parada>
        <llegada>9:00</llegada>
        <salida>20:00</salida>
      </escala>
      <escala dia="5">
        <parada>Bari</parada>
        <llegada>8:00</llegada>
        <salida>14:00</salida>
      </escala>
      <escala dia="6">
        <parada>Venecia</parada>
        <llegada>8:30</llegada>
        <salida></salida>
      </escala>
    </escalas>
  </crucero>
  <crucero codigo="CRUATL22">
    <destino>Atlántico (España, Marruecos, Portugal)</destino>
    <detalles>
      <cia>MSC Cruceros</cia>
      <dias>7 días</dias>
      <fechaSalida>2019-02-13</fechaSalida>
    </detalles>
    <escalas>
      <escala dia="1">
        <parada>Barcelona</parada>
        <llegada></llegada>
        <salida>18:00</salida>
      </escala>
      <escala dia="2">
        <parada>Navegación</parada>

```



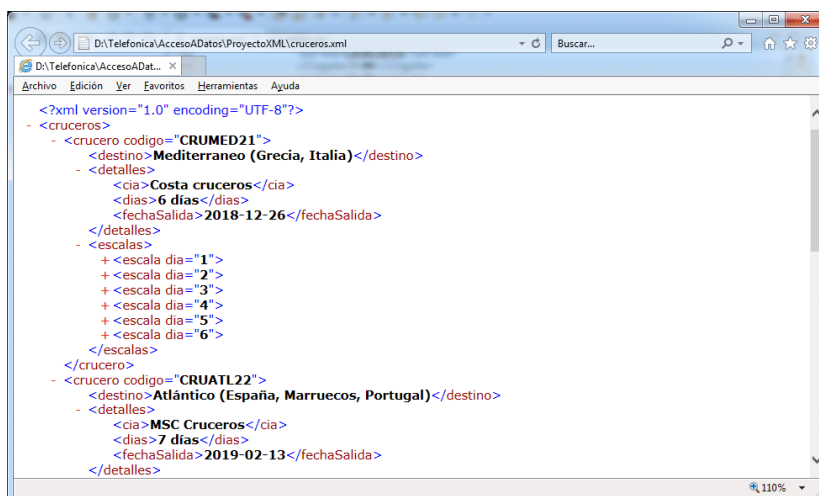
```

        <llegada></llegada>
        <salida></salida>
    </escala>
    <escala dia="3">
        <parada>Casablanca</parada>
        <llegada>7:00</llegada>
        <salida>22:00</salida>
    </escala>
    <escala dia="4">
        <parada>Navegación</parada>
        <llegada></llegada>
        <salida></salida>
    </escala>
    <escala dia="5">
        <parada>Santa Cruz de Tenerife</parada>
        <llegada>9:00</llegada>
        <salida>16:00</salida>
    </escala>
    <escala dia="6">
        <parada>Funchal</parada>
        <llegada>8:00</llegada>
        <salida>19:00</salida>
    </escala>
    <escala dia="7">
        <parada>Barcelona</parada>
        <llegada>17:00</llegada>
        <salida></salida>
    </escala>
</escalas>
</crucero>
</cruceros>

```

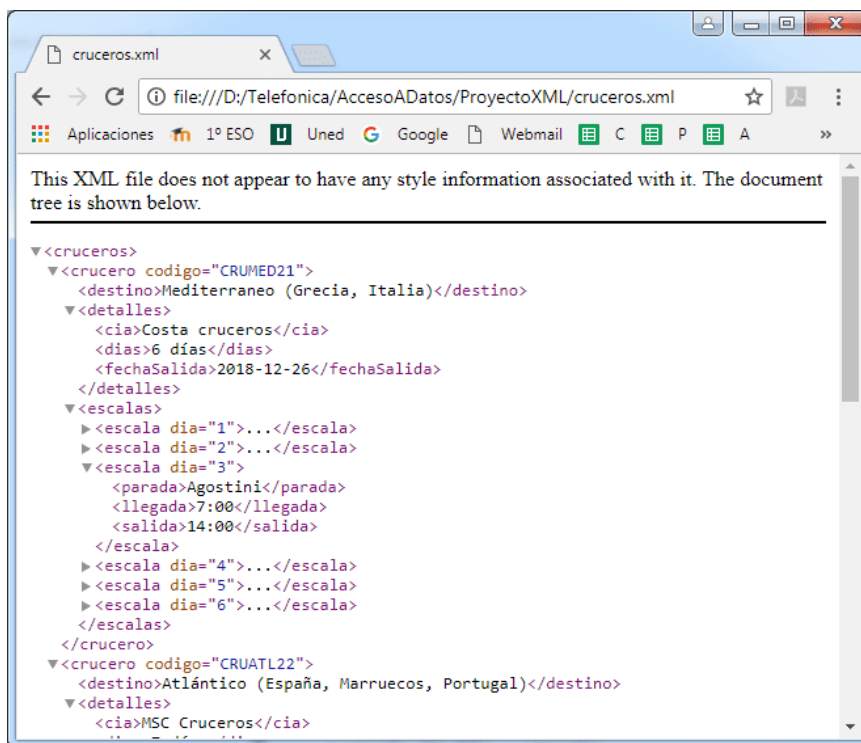
Con ayuda del explorador de archivos de Windows, accede a la carpeta del proyecto y abre el archivo *cruceros.xml* desde Internet Explorer.

Para lograrlo, haz clic derecho sobre el nombre del archivo en el explorador de Windows y luego selecciona "**Abrir con / Internet Explorer**".

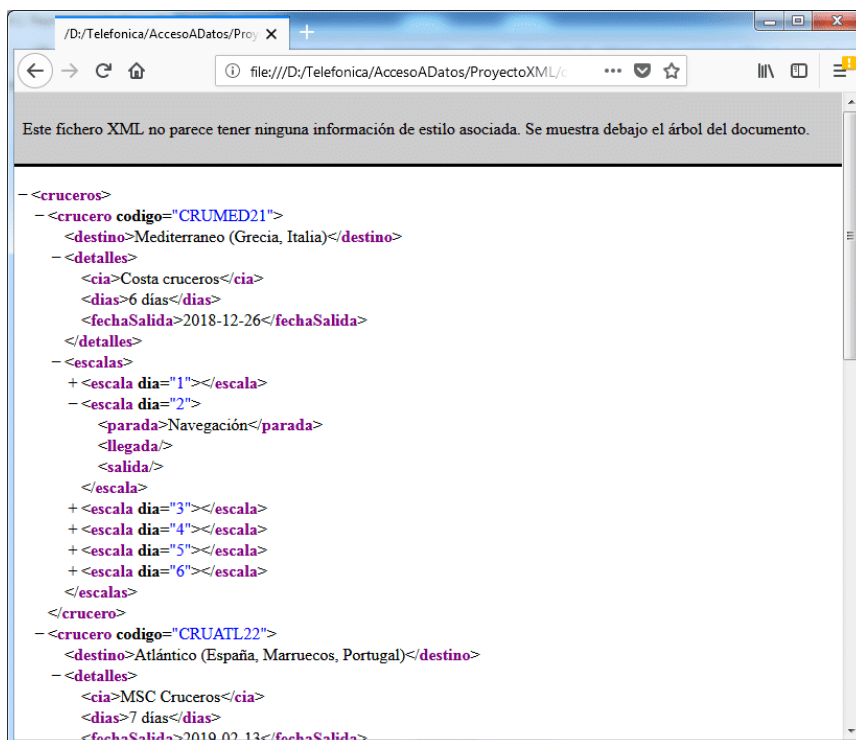


Vista de un documento XML con Internet Explorer. Delante de los nodos compuestos puedes apreciar que aparecen los signos - (contraer) y + (expandir).

Repite la operación con los navegadores que tengas instalados en tu equipo:



Vista de un documento XML con Google Chrome. Observa cómo delante de los nodos compuestos están los iconos de flecha hacia abajo (contraer) y flecha hacia la derecha (expandir).



Vista de un documento XML con Firefox. Igual que en Internet Explorer, puedes apreciar que aparecen los signos - (contraer) y + (expandir).

# Aplicar estilo al documento XML

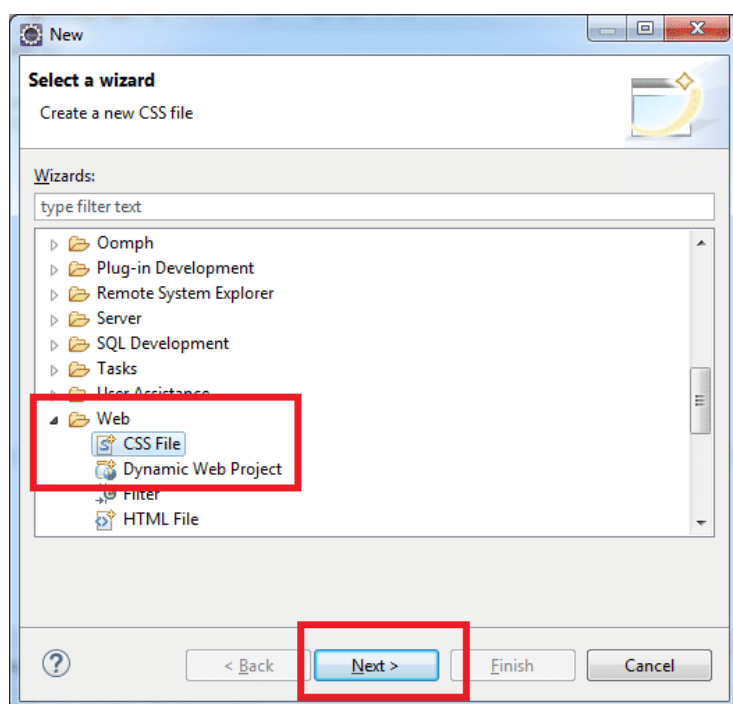
## Aplicar hojas de estilo CSS

En este apartado aprenderás a **aplicar diseño estético al contenido de tu documento XML** por medio de una hoja de estilo CSS.

Sigue estos pasos:

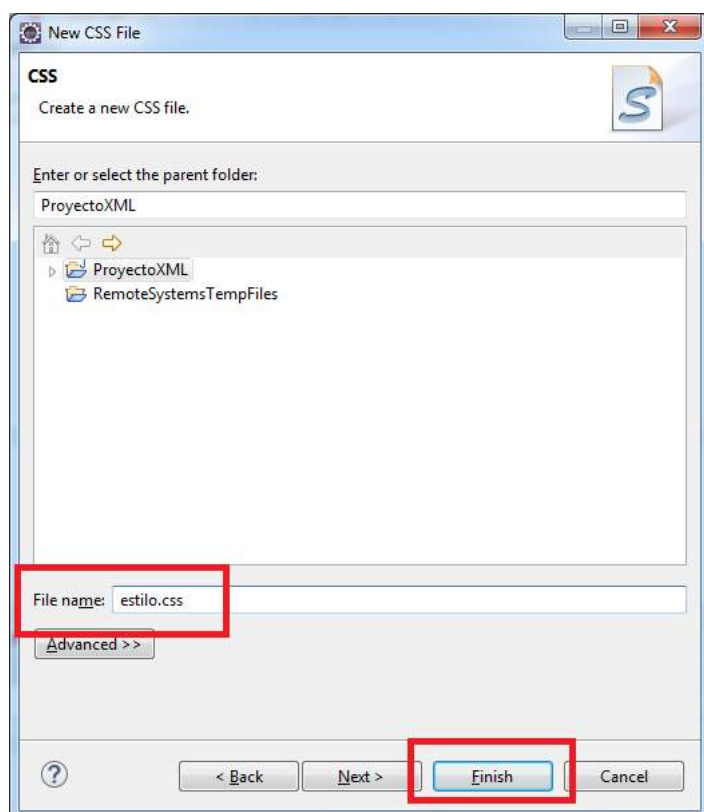
### 1. Crea el archivo de hoja de estilo en el mismo proyecto donde tienes el documento XML

- Abre el proyecto Eclipse que contiene el archivo *cruceros.xml* si no lo tienes ya abierto.
- Haz clic derecho sobre el nombre del proyecto y selecciona en el menú contextual *New / Other*.
- Ahora te encuentras en el cuadro de diálogo *New*. Abre la carpeta *Web*, selecciona *CSS File* y luego pulsa el botón *Next*.



- Escribe *estilo.css* como nombre de archivo y pulsa *Finish*.

#### 4.1. Representación de datos XML (CSS, XSL)



- Ahora, asigna un estilo a cada una de las etiquetas que forman parte del documento *cruceros.xml*. Copia y pega el código CSS que encontrarás a continuación.

```
@charset "ISO-8859-1";

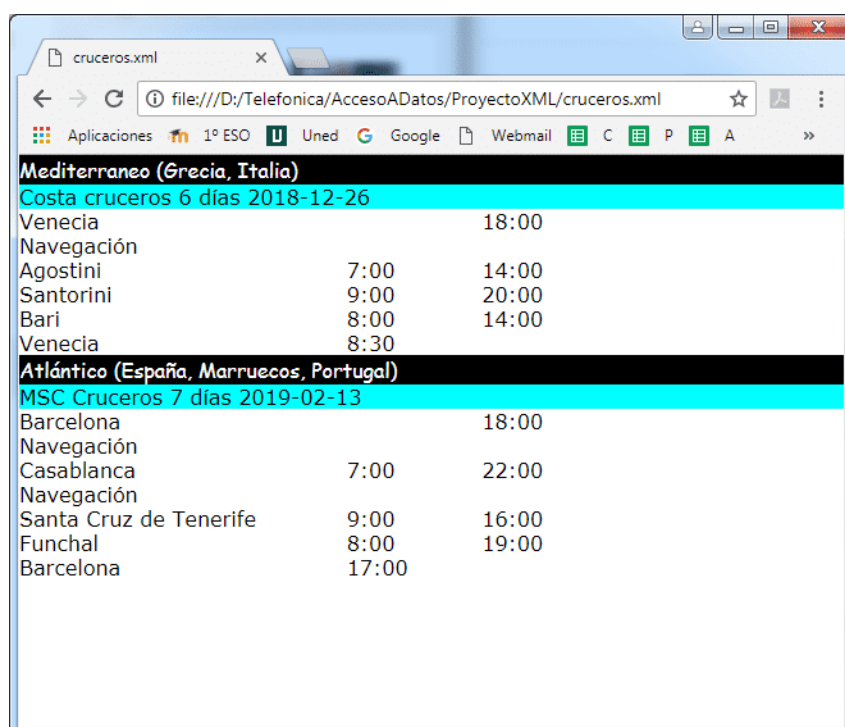
destino {
    background-color: black;
    color: white;
    font-family: "Comic sans ms";
    font-size: 30;
    display: block;
}
detalles {
    background-color: cyan;
    font-family: "Verdana";
    font-size: 20;
    display: block;
}
escala {
    font-family: "Verdana";
    font-size: 15;
    display: block;
}
parada {
    width: 250px;
    display: inline-block;
}
llegada, salida {
    width: 100px;
    display: inline-block;
}
```

## 2. Enlaza el archivo *cruceros.xml* con la hoja de estilos

- Para enlazar un documento XML con un archivo de hoja de estilos, se añade la etiqueta especial `<?xml-stylesheet ....>`, tal como verás a continuación.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo.css" ?>
<cruceros>
.....
.....
```

Ya puedes abrir tu documento *cruceros.xml* con cualquier navegador. El resultado será algo así:



Ya sabes cómo aplicar hojas de estilo CSS. ¿Continuamos con una nueva técnica?

## Trasformaciones XSL

A continuación vamos a mostrarte la técnica denominada **Trasformaciones XSL (*Extensible Stylesheet Language*)**.

Esta técnica está formada por un conjunto de tres lenguajes. Juntos son capaces de transformar la información contenida en los archivos XML para que pueda ser presentada al usuario en otros formatos, como, por ejemplo, HTML o PDF.

- **XSLT (*Extensible Stylesheet Language Transformations*)**: permite convertir documentos XML a otros formatos (por ejemplo, HTML).

#### 4.1. Representación de datos XML (CSS, XSL)

- **XSL-FO (XSL Formatting Objects):** permite especificar cómo se presentarán los datos, es decir, el diseño. Se utiliza principalmente para generar documentos PDF a partir de documentos XML.
- **XPath (XML Path Language):** permite filtrar la información contenida en un documento XML.

Los documentos de transformación XSL se guardan en archivos con extensión .xsl y tienen la estructura de una página web HTML, pero encerrada dentro de una plantilla como la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <!-- AQUÍ VA LA ESTRUCTURA DEL DOCUMENTO HTML -->
  </xsl:template>
</xsl:stylesheet>
```

Plantilla de un documento de transformación XSL.

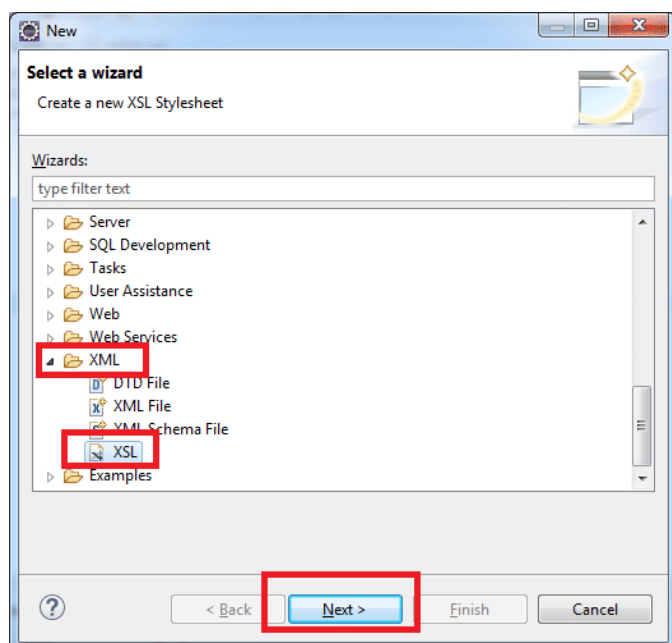
La estructura del documento HTML combina dos tipos de etiquetas:

- **Las etiquetas HTML**, tales como `<html>`, `<head>`, `<p>`, `<form>`, `<h1>`, etc.
- **Etiquetas especiales XSLT**, tales como `<xsl:value ....>` o `<xsl:for-each ....>`.

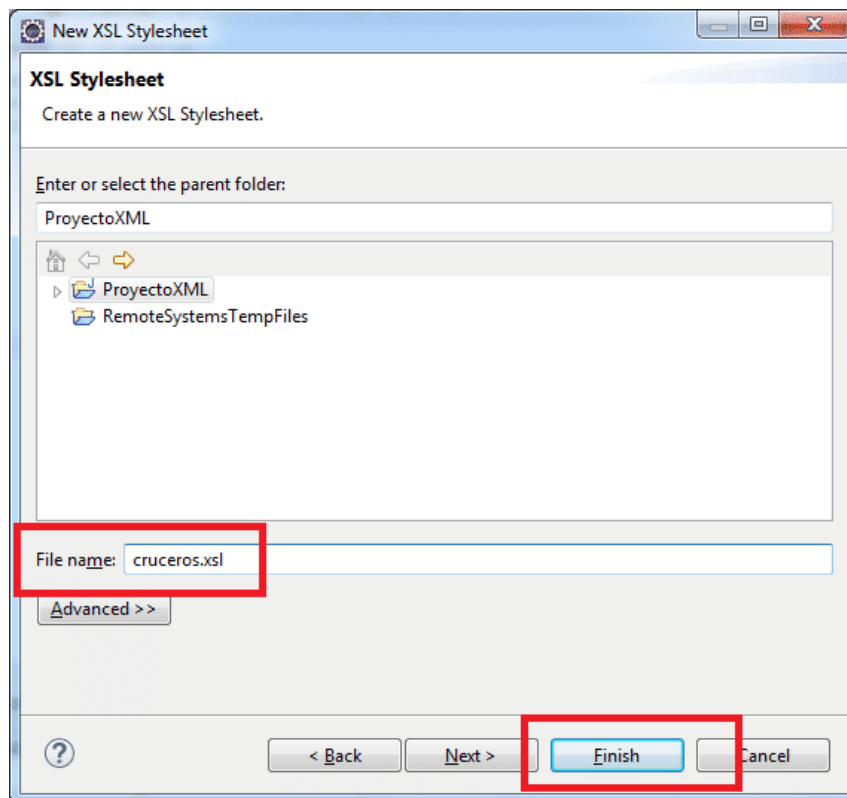
Ahora vas a crear tu primer archivo de transformación XSL y enlazarlo con el documento XML para dotarlo de estilo.

#### 1. Crea el archivo *cruceros.xsl* para formatear el documento *cruceros.xml*

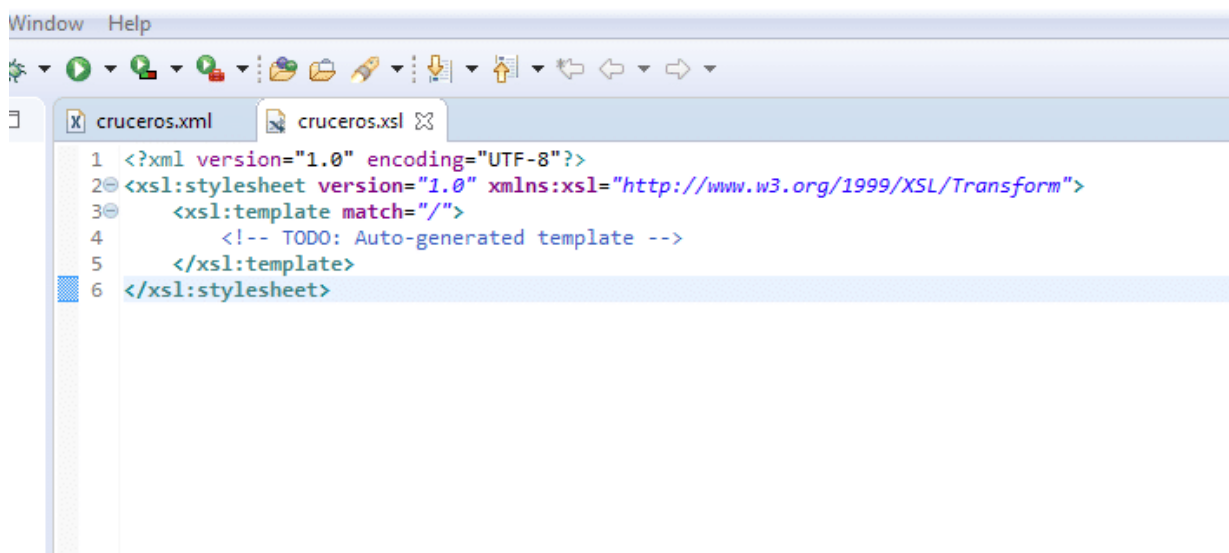
Haz clic derecho en el nombre del proyecto y selecciona en el menú contextual **New / Other**.



En el cuadro de diálogo *New*, abre la carpeta ***XML***, selecciona la opción ***XSL*** y pulsa el botón ***Next***.



Escribe ***cruceros.xml*** como nombre de archivo y pulsa el botón ***Finish***. Ya tienes la estructura de tu archivo de transformación XSL.



## 2. Edita el archivo ***cruceros.xml***

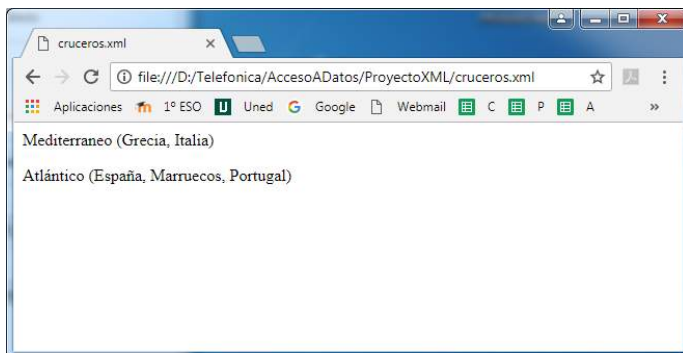
Como ves, ya tienes la estructura básica del archivo xsl. Vamos ahora a comenzar por un sencillo ejemplo que muestre el destino de los cruceros.



**Edita el archivo cruceros.xsl hasta dejarlo como se muestra a continuación:**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Cruceros</title>
      </head>
      <body>
        <xsl:for-each select="cruceros/crucero">
          <p>
            <xsl:value-of select="destino"/>
          </p>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Este es el resultado que esperamos obtener, una vez que hayamos enlazado el documento XML con el archivo de transformación XSL:



Vamos a analizar el código.

En primer lugar, dentro de las etiquetas **<xsl:template match="/"> ... </xsl:template>** está encerrada toda la estructura de la página web que queremos visualizar en el navegador.

Pero queremos iterar dentro del documento XML cada una de las etiquetas o elementos *crucero* para mostrar el *destino*. Para iterar o recorrer todos los elementos *crucero*, necesitamos una etiqueta de tipo **<xsl:for-each ..... </xsl:for-each>**.

```
<xsl:for-each select="cruceros/crucero">
  <p>
    <xsl:value-of select="destino"/>
  </p>
</xsl:for-each>
```

Por cada uno de los cruceros iterados, por medio de la etiqueta **<xsl:value-of select="destino"/>**, estamos mostrando el elemento XML *destino*.

### 3. Enlaza el documento **cruceros.xml** con el archivo de transformación **cruceros.xsl**.

Para enlazar el documento XML con el archivo de transformación, debes añadir justo delante de la etiqueta raíz un elemento de tipo **<?xml-stylesheet .....>**. Edita el documento **cruceros.xml** para añadir la segunda línea que ves en este código:

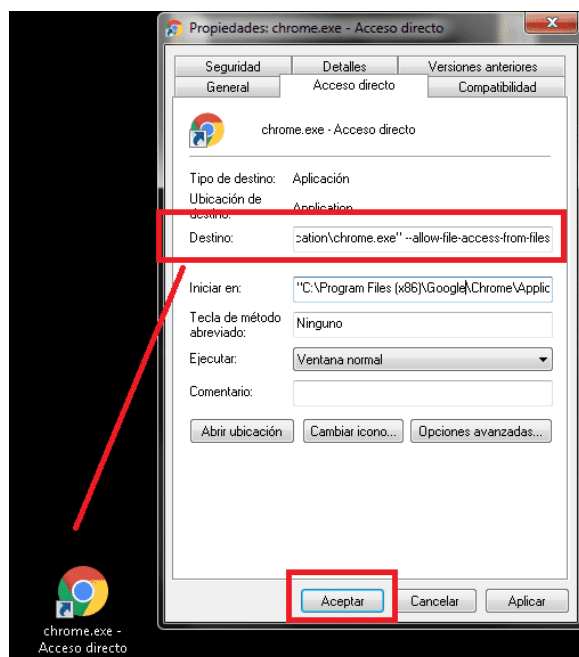
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cruceros.xsl"?>
<cruceros>
```

La nueva línea añadida sustituirá a la anterior, que enlazaba con el archivo CSS.

### 4. Visualiza el resultado en tu navegador

**Abre el documento **cruceros.xml**** con Internet Explorer o Mozilla Firefox. Si intentas abrirlo desde Google Chrome en modo local, no funcionará. Esto se debe a problemas de configuración de la seguridad, ya que no permite abrir el archivo xsl en local. Sin embargo, sí funcionará cuando esté subido a un *hosting* web y accedas a **cruceros.xml** en modo remoto. Podría funcionar en modo local con Chrome si abres la aplicación con el parámetro **--allow-file-access-from-files**.

Para hacer la prueba, tendrías que crear un acceso directo a la aplicación **chrome.exe**, hacer clic derecho sobre el acceso directo, seleccionar **Propiedades** en el menú contextual y añadir al final del campo destino el parámetro **--allow-file-access-from-files**. Por último, pulsamos el botón **Aceptar** y podremos visualizar nuestro documento XML, siempre y cuando hayamos abierto Chrome desde el acceso directo.



**Completa el código para mostrar toda la información posible de cada crucero**

Edita de nuevo el archivo *cruceros.xsl* hasta dejarlo del siguiente modo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
    <head>
    <title>Cruceros</title>
    </head>
    <body>
    <xsl:for-each select="cruceros/crucero">
      <p>
        Destino: <xsl:value-of select="destino"/>
        <br />
        Dias: <xsl:value-of select="detalles/cia"/>
        <br />
        Dias: <xsl:value-of select="detalles/dias"/>
        <br />
        Fecha salida: <xsl:value-of
select="detalles/fechaSalida"/>
      </p>
      <table>
        <tr>
          <th>Dia</th>
          <th>Parada</th>
          <th>Llegada</th>
          <th>Salida</th>
        </tr>
        <xsl:for-each select="escalas/escala">
          <tr>
            <td><xsl:value-of select="@dia"/></td>
            <td><xsl:value-of select="parada"/></td>
            <td><xsl:value-of select="llegada"/></td>
            <td><xsl:value-of select="salida"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </xsl:for-each>
    </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```

Ahora, el resultado visualizado en el navegador quedará así:

Destino: Mediterraneo (Grecia, Italia)  
Días: Costa cruceros  
Días: 6 días  
Fecha salida: 2018-12-26

Día	Parada	Llegada	Salida
1	Venecia		18:00
2	Navegación		
3	Agostini	7:00	14:00
4	Santorini	9:00	20:00
5	Bari	8:00	14:00
6	Venecia	8:30	

Destino: Atlántico (España, Marruecos, Portugal)  
Días: MSC Cruceros  
Días: 7 días  
Fecha salida: 2019-02-13

Día	Parada	Llegada	Salida
1	Barcelona		18:00
2	Navegación		
3	Casablanca	7:00	22:00
4	Navegación		
5	Santa Cruz de Tenerife	9:00	16:00
6	Funchal	8:00	19:00
7	Barcelona	17:00	

Vamos a analizar el nuevo código

```
<xsl:for-each select="cruceros/crucero">
  <p>
    Destino: <xsl:value-of select="destino"/>
    <br />
    Días: <xsl:value-of select="detalles/cia"/>
    .....
  </p>
  <table>
    .....
    <xsl:for-each select="escalas/escala">
      <tr>
        <td><xsl:value-of select="@dia"/></td>
        <td><xsl:value-of select="parada"/></td>
        <td><xsl:value-of select="llegada"/></td>
        <td><xsl:value-of select="salida"/></td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:for-each>
```

## 4.1. Representación de datos XML (CSS, XSL)

En primer lugar, utilizamos la siguiente estructura:

```
<xsl:for-each select="cruceros/crucero">
```

```
.....
```

```
</xsl:for-each>
```

Esto representa una estructura de control repetitiva que se ejecuta para cada elemento *crucero* dentro de otro elemento *cruceros*. En cada repetición se establece un filtro de modo, que delimita el contenido XML al elemento *crucero* que toque en cada iteración.

**Destino:** **<xsl:value-of select="destino"/>**

**Días:** **<xsl:value-of select="detalles/cia"/>**

Estas etiquetas permiten mostrar el *destino* y *cia* que corresponde al *crucero* de la iteración actual. Observa que en el atributo *select* se especifica el elemento a mostrar como referencia relativa a partir de *cruceros/crucero* (*select* especificado en la estructura *for-each*).

```
<xsl:for-each select="cruceros/crucero">
```

```
  <xsl:for-each select="escalas/escala">
```

```
    <tr>
```

```
      <td><xsl:value-of select="@dia"/></td>
```

```
      <td><xsl:value-of select="parada"/></td>
```

```
      <td><xsl:value-of select="llegada"/></td>
```

```
      <td><xsl:value-of select="salida"/></td>
```

```
    </tr>
```

```
  </xsl:for-each>
```

```
</xsl:for-each>
```

El *for-each* interno está estableciendo filtro sobre filtro. Por cada elemento *crucero* dentro de *cruceros*, itera sus elementos *escala* dentro de *escalas* para mostrar los detalles de cada escala. El carácter *@* colocado dentro del atributo *select* identifica a un atributo de una etiqueta, no a una etiqueta, es decir, hace referencia al atributo *dia* de la etiqueta *escala* (*<escala dia="1">*).

Aún podríamos mejorar el aspecto estético del documento, enlazándolo con un archivo de hoja de estilos CSS. ¿Quieres aprender cómo se hace?

## Usar XML, XSL y CSS

Hemos transformado el contenido XML para representarlo en el navegador dentro de una estructura HTML, pero todavía podríamos darle un estilo mucho más estético.

Para lograrlo, vamos a aplicar también CSS.

**Pero, ¿qué nos aporta XSL? ¿No podríamos usar sólo CSS, como en el primer ejemplo?**

XSL nos brinda la posibilidad de filtrar la información contenida en el archivo XML, algo que no es posible sólo con CSS.

Lo verás con más claridad en el apartado siguiente (*XPath*).

Sabemos que un archivo de transformación XSL encierra dentro de una plantilla la estructura de una página HTML con sus típicas etiquetas `<html>`, `<head>`, `<body>`, etc. Para utilizar también una hoja de estilo CSS, lo único que tenemos que hacer es usar una etiqueta de tipo `<link rel="stylesheet" .....>` dentro de dicha página.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Cruceros</title>
        <link rel="stylesheet" href="css/estilo.css" />
      </head>
      <body>
        .....
```

En el código anterior, hemos incluido la siguiente línea dentro de la cabecera de la página:  
`<link rel="stylesheet" href="css/estilo.css" />`

Ahora, sólo tienes que construir el archivo `estilo.css` dentro de una carpeta `css` con los atributos que desees.

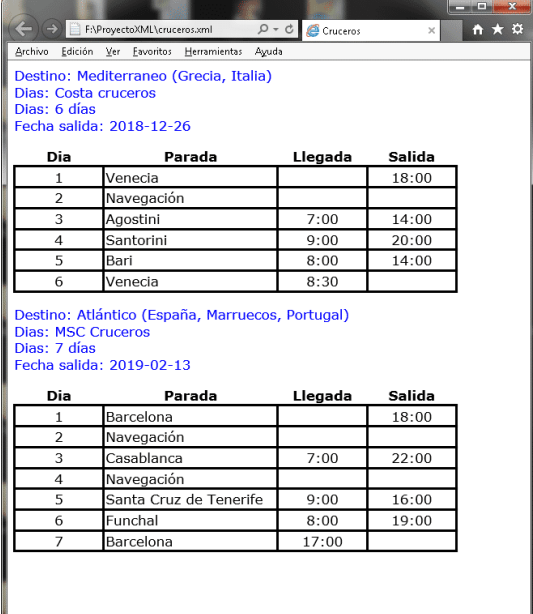
```
@CHARSET "ISO-8859-1";
body {
    font-family: Verdana;
}
p {
    color: blue;
}
td {
    width: 100px;
    text-align: center;
    border: solid;
}
table {
    border-collapse: collapse;
}
td.masAncho {
    width: 200px;
    text-align: left;
}
```

## 4.1. Representación de datos XML (CSS, XSL)

Además de aplicar atributos a las etiquetas HTML *body*, *p*, *td* y *table*, hemos creado una clase CSS llamada *masAncho*, aplicable a elementos de tipo *td* (celda de tabla). Eso te obliga a editar de nuevo el archivo de transformación XSL para modificar una simple línea:

```
<td class="masAncho"><xsl:value-of select="parada"/></td>
```

Si has seguido bien todos los pasos, el resultado en el navegador será el siguiente:



Destino: Mediterráneo (Grecia, Italia)  
Días: Costa cruceros  
Días: 6 días  
Fecha salida: 2018-12-26

Día	Parada	Llegada	Salida
1	Venecia		18:00
2	Navegación		
3	Agostini	7:00	14:00
4	Santorini	9:00	20:00
5	Bari	8:00	14:00
6	Venecia	8:30	

Destino: Atlántico (España, Marruecos, Portugal)  
Días: MSC Cruceros  
Días: 7 días  
Fecha salida: 2019-02-13

Día	Parada	Llegada	Salida
1	Barcelona		18:00
2	Navegación		
3	Casablanca	7:00	22:00
4	Navegación		
5	Santa Cruz de Tenerife	9:00	16:00
6	Funchal	8:00	19:00
7	Barcelona	17:00	

## Filtrar los resultados con XPath

**XPath** forma parte de la sintaxis de los documentos XSL, y tiene como objetivo buscar y seleccionar partes de un documento XML dentro de su estructura jerárquica.

En nuestro archivo de transformación XSL utilizamos la siguiente línea para iterar los distintos elementos *<crucero>* dentro de la estructura del documento XML:

```
<xsl:for-each select="cruceros/crucero">
```

Dentro del atributo *select* estás seleccionando todos los elementos *crucero* que están dentro de una etiqueta padre llamada *cruceros*. Sin saberlo, ya estás aplicando la tecnología **XPath**.

**<xsl:for-each select="cruceros/crucero">**



**Esto es XPath**

XPath se aplica en el valor de los atributos *select* de las etiquetas XSL.



## 4.1. Representación de datos XML (CSS, XSL)

Hasta ahora hemos utilizado la versión más simple de XPath, pero su sintaxis cuenta con caracteres especiales que brindan un sistema muy potente para la búsqueda de datos en documentos XML. Vamos a ver varios ejemplos:

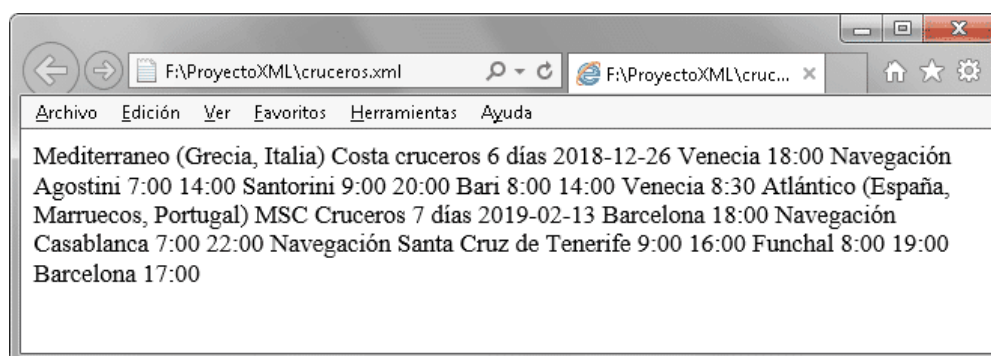
### 1. El carácter \ selecciona todo el contenido del elemento raíz.

Deja tu archivo de transformación XSL así de sencillo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
    <head>
    <title>Cruceros</title>
    </head>
    <body>
      <xsl:value-of select="\ " />
    </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```



Como resultado ofrece todo el texto contenido dentro de la etiqueta raíz, que, para nuestro documento, es la etiqueta **<cruceros>**.

### 2. Los caracteres // seleccionan todos los elementos situados en cualquier nivel.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
    <head>
    <title>Cruceros</title>
    </head>
    <body>
      <xsl:for-each select="//escala">
        <xsl:value-of select="." /><br />
      </xsl:for-each>
    </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```

Seleccionaría todos los elementos *escala*, independientemente de la posición en la que se encuentren dentro del árbol jerárquico del documento XML.

En nuestro documento XML, todos los elementos *escala* están al mismo nivel, con lo que no notarás la diferencia si pones *cruceros/crucero/escalas/escala*. Puesto que existen varios elementos *escala*, es necesario iterar con una estructura *for-each*. En el *select* interno, el punto que aparece hace referencia al contenido de cada elemento iterado.



Resultado.

### 3. Podemos especificar entre corchetes [ ] el elemento que queremos seleccionar dentro de una colección de ellos.

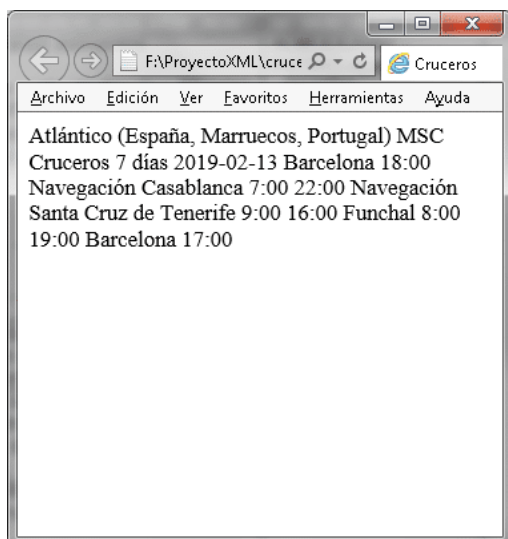
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
      <head>
        <title>Cruceros</title>
      </head>
      <body>
        <xsl:value-of select="cruceros/crucero[2]" /><br />
      </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```

Selecciona y muestra sólo el segundo crucero.

## 4.1. Representación de datos XML (CSS, XSL)



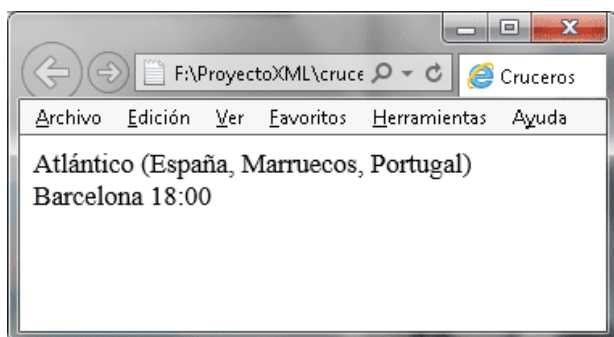
Resultado.

**4. Ahora queremos seleccionar el destino del segundo crucero y el puerto de donde sale, es decir, la primera escala del segundo crucero.**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
      <head>
        <title>Cruceros</title>
      </head>
      <body>
        <xsl:value-of select="cruceros/crucero[2]/destino" /><br />
        <xsl:value-of select="cruceros/crucero[2]/escalas/escala[1]"
/><br />
      </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```



Resultado.

**5. También podemos utilizar la función *last()* dentro de los corchetes.**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
    <head>
    <title>Cruceros</title>
    </head>
    <body>
      <xsl:value-of select="cruceros/crucero[last()]/destino" /><br />
      <xsl:value-of select="cruceros/crucero[last()]/escalas/escala[1]" /><br />
    </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```

Ofrece el mismo resultado que el ejemplo anterior. Muestra el destino del último crucero, que en nuestro caso es el segundo, y luego muestra la primera escala.

**6. También podemos establecer criterios de búsqueda utilizando los atributos de las etiquetas XML. En nuestro documento tenemos los atributos *codigo* de crucero y *dia* en las escalas.**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">

    <html>
    <head>
    <title>Cruceros</title>
    </head>
    <body>
      <xsl:value-of
select="cruceros/crucero[@codigo='CRUATL22']/destino" /><br />
      <xsl:value-of
select="cruceros/crucero[@codigo='CRUATL22']/escalas/escala[1]" /><br />
    </body>
    </html>

  </xsl:template>
</xsl:stylesheet>
```

Este ejemplo ofrece el mismo resultado que el anterior, pero filtra los cruceros por el atributo *codigo*.

**¿Quieres saber más sobre XPath?**

Puedes consultar el tutorial de XPath de W3Schools haciendo clic en el botón de la derecha.

[https://www.w3schools.com/xml/xpath\\_intro.asp](https://www.w3schools.com/xml/xpath_intro.asp)

# Despedida

## Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

- Los ficheros **XML** (***Extended Markup Language***) proveen un formato ligero para el intercambio de datos, lo que resulta especialmente interesante en aplicaciones web.
- **HTML se basa en el estándar SGML** que sigue un esquema jerárquico compuesto por etiquetas con apertura y cierre que tienen la siguiente estructura: `<etiqueta> contenido </etiqueta>`. Cada etiqueta en un documento XML representa un nodo o elemento, que puede estar compuesto por otros nodos.
- Es posible formatear la presentación de los documentos XML en el navegador web por medio de **hojas de estilo CSS**. Podemos aplicar atributos de estilo a las distintas etiquetas, pero no es posible aplicar filtros.
- La técnica denominada **Trasformaciones XSL** (***EXtensible Stylesheet Language***) está formada por un conjunto de tres lenguajes que juntos son capaces de transformar la información contenida en los archivos XML para que pueda ser presentada al usuario en otros formatos, por ejemplo, HTML o PDF. Estos tres sublenguajes son **XSLT**, **XSL-FO** y **XPath**.
- Usando un archivo de transformación XSL podemos **establecer filtros dentro del fichero XML** para seleccionar exactamente la información que queremos presentar al usuario.