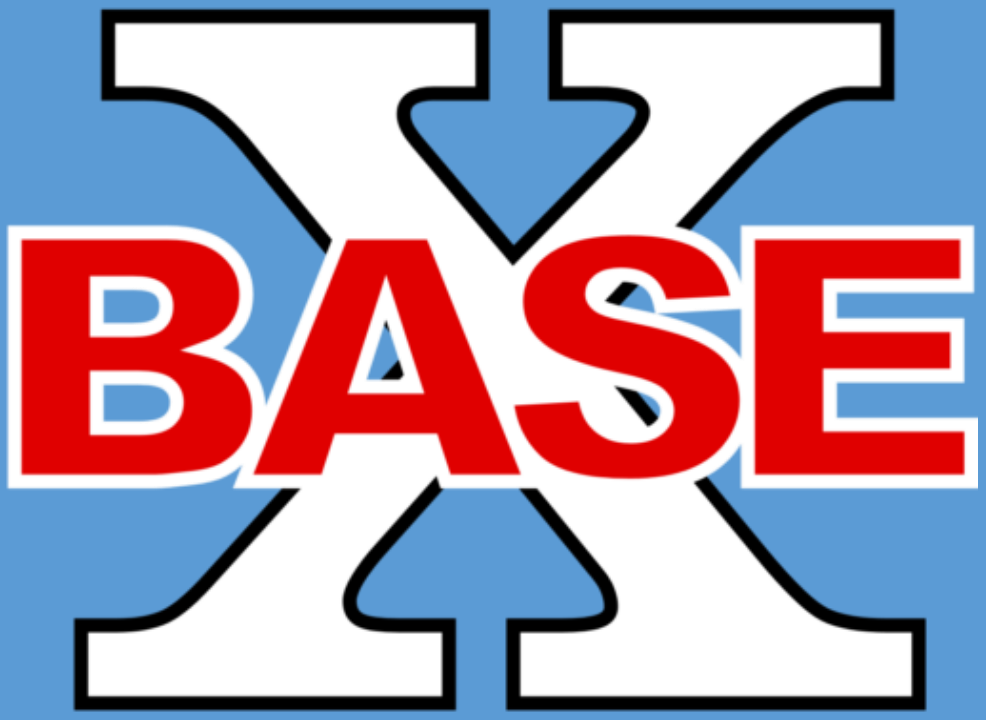


Cora Perez Manivesa
1ºASIR

INDICE

Introducción

Que es Base X?

"BaseX es un motor de base de datos XML ligero, de alto rendimiento y escalable y XPath / XQuery 3.1 , que incluye soporte completo para las extensiones de actualización y texto completo del W3C. Una interfaz gráfica de usuario interactiva y fácil de usar, que le da una gran visión de sus documentos XML."

INSTALACIÓN

Lo Primero que tenemos que hacer es descargar el programa desde aqui .

Segundo descargamos el fichero

Tercero movemos el directorio basex al directorio donde lo queramos guardar si no queremos usar el directorio que usa el por defecto.

Terceo ejecutamos el BaseX.jar

INTEGRACION DE BASEX EN ECLIPSE

BASEX es una poderosa herramienta para desarrollar e implementar aplicaciones basadas en XML, y Eclipse IDE es uno de los entornos de desarrollo integrado (IDE) más populares para aplicaciones basadas en Java. La integración de BASX en Eclipse IDE puede ayudar a los desarrolladores a

optimizar su flujo de trabajo y mejorar la productividad al proporcionar un amplio conjunto de herramientas y características para el desarrollo de XML.

Para integrar BASX en Eclipse IDE, se deben seguir algunos pasos. El primer paso es abrir Eclipse IDE y hacer clic en "Ayuda" en la barra de menú. Desde allí, seleccione "Eclipse Marketplace" en el menú desplegable.

En la ventana de Eclipse Marketplace, puede buscar "BASX" en la barra de búsqueda y seleccionar el complemento correspondiente. Haga clic en el botón "Instalar" para instalar el complemento BASX. Una vez completada la instalación, reinicie Eclipse IDE.

Después de reiniciar Eclipse IDE, ahora puede usar BASX en Eclipse IDE para desarrollar y depurar sus aplicaciones basadas en XML. BASX proporciona una serie de funciones y herramientas que pueden ayudar a los desarrolladores a trabajar de manera más eficiente, que incluyen:

Editor de esquemas XML: este editor proporciona una vista completa de la estructura de un documento XML y permite a los desarrolladores crear y editar esquemas XML fácilmente.

Editor XML: este editor proporciona una interfaz fácil de usar para crear y editar documentos XML, con resaltado de sintaxis y otras funciones que facilitan la escritura de código sin errores.

Editor XSLT: este editor proporciona una interfaz visual para crear y editar hojas de estilo XSLT, que se utilizan para transformar documentos XML en otros formatos.

Editor XPath: este editor proporciona una poderosa herramienta para trabajar con expresiones XPath, que se utilizan para navegar y seleccionar elementos y atributos en un documento XML.

Depurador XML: este depurador permite a los desarrolladores recorrer aplicaciones basadas en XML e identificar errores y otros problemas.

Además de estas características, BASX también brinda soporte para otras tecnologías relacionadas con XML, como XML Signature y XML Encryption, lo que la convierte en una poderosa herramienta para desarrollar aplicaciones seguras y confiables basadas en XML.

La integración de BASX en Eclipse IDE puede ayudar a mejorar la productividad y reducir el tiempo y el esfuerzo necesarios para desarrollar e implementar aplicaciones basadas en XML. Con sus potentes funciones y herramientas, BASX es una herramienta esencial para cualquier desarrollador que trabaje con tecnologías basadas en XML, e integrarlo con Eclipse IDE es un proceso simple y directo que puede brindar beneficios significativos a cualquier equipo de desarrollo.ECLIPSE

3.b)TRANSFORMACIÓN EL XML A JSON

BAS E X es una poderosa herramienta para transformar documentos XML a otros formatos, incluido JSON. Transformar un documento XML en JSON puede ser útil en una variedad de contextos, como cuando se trabaja con API web que requieren datos JSON o cuando se integra con otros sistemas que usan JSON como su formato de datos principal.

Para transformar un documento XML en JSON utilizando BAS E X, se deben seguir algunos pasos:

Abra BAS E X y cree una nueva hoja de estilo XSLT. XSLT es un lenguaje poderoso para transformar documentos XML en otros formatos, y BAS E X proporciona una serie de herramientas y funciones para trabajar con XSLT.

En la hoja de estilo XSLT, cree una plantilla que coincida con el elemento raíz del documento XML. Esta plantilla definirá la estructura general del documento JSON.

Dentro de la plantilla del elemento raíz, cree plantillas adicionales para cada elemento o atributo que deba incluirse en el documento JSON. Estas plantillas definirán cómo se deben transformar los datos XML en datos JSON.

Utilice las instrucciones "xsl:element" y "xsl:value-of" para crear la estructura de datos JSON. La instrucción "xsl:element" crea un nuevo elemento JSON, mientras que la instrucción "xsl:value-of" inserta el valor de un elemento o atributo XML en el elemento JSON.

Utilice la instrucción "xsl:if" para incluir o excluir condicionalmente elementos del documento JSON. Esto puede ser útil para filtrar datos innecesarios o para manejar casos extremos.

Guarde la hoja de estilo XSLT y ejecútela en el documento XML mediante BAS E X. El resultado será un documento JSON que refleje la estructura y el contenido del documento XML original.

En general, transformar un documento XML en JSON mediante BAS E X es un proceso sencillo que se puede lograr mediante plantillas e instrucciones XSLT. Al aprovechar el poder y la flexibilidad de XSLT, los desarrolladores pueden convertir fácilmente los datos XML en un formato ampliamente utilizado y compatible en el panorama moderno del desarrollo web.

BAS E X proporciona otras características y herramientas para trabajar con XML y XSLT, incluida la validación de esquemas, la evaluación de XPath y la generación de documentos XML. Con su amplio conjunto de características y poderosas capacidades de transformación, BAS E X es una herramienta valiosa para cualquier desarrollador que trabaje con datos XML y puede ayudar a optimizar el proceso de desarrollo y mejorar la productividad.

Supongamos que tenemos un archivo XML con información sobre algunos productos. Aquí está el contenido del archivo XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <product>
    <name>Product A</name>
    <price>10.99</price>
    <category>Category A</category>
  </product>
  <product>
    <name>Product B</name>
    <price>25.99</price>
    <category>Category B</category>
  </product>
  <product>
    <name>Product C</name>
    <price>5.99</price>
    <category>Category A</category>
  </product>
</products>
```

Para transformar este archivo xml a json debe tener la siguiente estructura

```
{
 "products": [
  {
   "name": "Product A",
   "price": 10.99,
   "category": "Category A"
  },
  {
   "name": "Product B",
   "price": 25.99,
   "category": "Category B"
  },
  {
   "name": "Product C",
   "price": 5.99,
   "category": "Category A"
  }
 ]
}
```

Para hacer esto seguiremos los pasos citados acontinuación:

Abrimos BAS E X y creamos un nuevo archivo XSLT. Llamémoslo "xml-to-json.xslt".

En el archivo XSLT, agregamos las siguientes líneas para definir el esqueleto del archivo JSON:

```
<
xsl:stylesheet

version
=
"1.0"

xmlns:xsl
```

=

"http://www.w3.org/1999/XSL/Transform"

>

<

xsl:template

match

=

"/"

>

<

xsl:text

>

{ "products": [

</

xsl:text

>

<

xsl:apply-templates

select

=

"products/product"

/>

<

```
xsl:text
>
] }
</
xsl:text
>



</
xsl:template
>
```

Esta plantilla nos permitirá establecer la estructura "xsl:text" se usa para agregar texto estático a la salida. La instrucción "xsl:apply-templates" se utiliza para procesar todos los elementos "product" del archivo XML.

3. A continuación, agregamos las siguientes líneas para procesar los elementos "product" del archivo XML:

```
<xsl:template match="product">
   <xsl:text>{ "name": "</xsl:text>
   <xsl:value-of select="name"/>
   <xsl:text>", "price": </xsl:text>
   <xsl:value-of select="price"/>
   <xsl:text>, "category": "</xsl:text>
   <xsl:value-of select="category"/>
   <xsl:text>" }</xsl:text>
   <xsl:if test="position()!=last()">
     <xsl:text>,</xsl:text>
   </xsl:if>
 </xsl:template>
```

3.a)DEVOLUCIÓN CONSULTAS XQUERY

En este apartado vamos a utilizar el xml de los libros para realizar dos consultas en XQUERY:

11) let $visitas:= /tutoriales/tutorial/visitas

return <media> {avg($visitas)}</media>

1) for $lascategorias in /tutoriales/tutorial/categoría return 4lascategorias

3.c)DEVOLUCIÓN DE CONSULTAS EN XPHAT:

Para este ejemplo vamos a utilizar el XML de libros para realizar las siguientes consultas:

a)/tutoriales/tutorial

//autor/nombre[string-length(text()) < 9]

3.D) Valida un documento contra un XML Schema

En este caso usaremos el xml de los tutoriales y haremos un xsd para validarlo.

A continuación vamos a explicar el código del XSD.

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

Establecemos el prefijo de namespace 'xs' y define que estamos creando una XML Schema.

<xs:element name="equipos">

<xs:complexType>

<xs:sequence>

...

</xs:sequence>

</xs:complexType>

</xs:element>

En estas líneas definimos el elemento raíz **'equipos'.** Su tipo complejo contiene una secuencia de elementos.

<xs:element name="máquina" maxOccurs="unbounded">

```
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
    <xs:attribute name="nome" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

El elemento **'máquina'** puede aparecer varias veces, por lo que **'maxOccurs'** se establece en 'unbounded'.Su tipo complejo contiene una secuencia de elementos u un atributo 'nome' de tipo '**string'** que es obligatorio

.

```
<xs:element name="hardware">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

El elemento **'hardware'** tiene tipo complejo que contiene una secuencia de elementos.

```
<xs:element name="tipo" type="xs:string"/>
```

El elemento 'tipo' tiene un tipo de 'string'.

```
<xs:element name="procesador">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="marca" type="xs:string" use="required"/>
```

```
      <xs:attribute name="num_nucleos" type="xs:integer" use="required"/>
      <xs:attribute name="velocidade" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

EL procesador 'procesador' tiene un tipo complejo con contenido simple.Es decir,su contenido es solo texto y tiene atributos de **'marca', 'num_nucleos' y 'velocidade'**.

```
<xs:element name="memoria">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:integer">
    <xs:attribute name="tecnoloxía" type="xs:string" use="required"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

El elemento **'memoria'** también tiene un tipo complejo con contenido simple y un atributo de **'tecnoloxías'.**

```
<xs:element name="disco" maxOccurs="unbounded">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="tecnoloxía" type="xs:string" use="required"/>
    <xs:attribute name="capacidade" type="xs:integer" use="required"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
```

</xs:element>

Lo que **'maxOccurs="unbounded"** indica que puede haber varios elementos '**disco'** dentro de '**hardware**'. También tiene un tipo complejo con contenido simple y dos atributos de''**tecnoloxía'** y '**capacidade**'.

```
<xs:element name="gravadora">
  <xs:complexType>
    <xs:attribute name="tipo" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

El elemento **'gravadora'** tiene un tipo complejo que solo tiene un atributo **'tipo'** requerido.

```
<xs:element name="config">
  <xs:complexType>
    <xs:sequence>
      ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

El elemento **'config'** tiene un tipo complejo que contiene una secuencia de elementos,

<xs:element name="OS" type="xs:string"/>

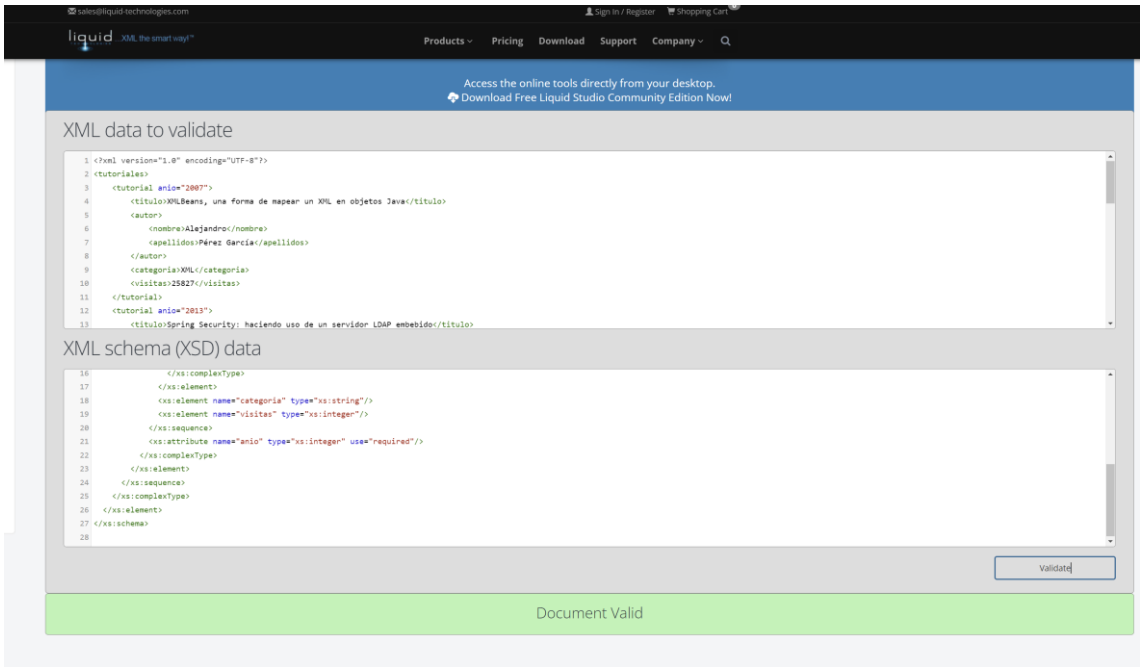
El elemento **OS** tiene un tipo simple de **string**.

<xs:element name="IP" type="xs:string"/>

El elemento' **IP'** tiene un tipo simple de '**string'.**

<xs:element name="gateway" type="xs:string"/>

El elemento **'gateway'** tiene un tipo simple de **'string'**.



e) Valida un documento contra un DTD (1 punto)

Para validar el XML anterior contra un DTD, primero debemos crear un archivo DTD que describa la estructura del XML. A continuación se muestra un posible archivo DTD que se puede usar para validar el XML anterior:

<!ELEMENT

equipos

(má

quina

*)>

<!ELEMENT máquina (

hardware

,
config
)>

<!ATTLIST máquina
nome

CDATA

#REQUIRED
>

<!ELEMENT
hardware
(
tipo
,
fabricante
,
procesador
,
memoria
,
disco
+,
gravadora
?)>

<!ELEMENT
tipo
(
#PCDATA
)>

<!ELEMENT
fabricante
(
#PCDATA
)>

<!ELEMENT
procesador
(
#PCDATA
)>

<!ATTLIST
procesador

marca

CDATA

#REQUIRED

num_nucleos

CDATA

#REQUIRED

velocidade

CDATA

#REQUIRED
>

<!ELEMENT
memoria
(
#PCDATA
)>

<!ATTLIST
memoria

tecnolox
ía
CDATA

#REQUIRED
>

<!ELEMENT
disco

EMPTY
>

<!ATTLIST
disco

tecnolox
ía
CDATA

#REQUIRED

capacidade

CDATA

#REQUIRED
>

<!ELEMENT
gravadora

EMPTY
>

<!ATTLIST
gravadora

tipo

CDATA

#REQUIRED
>

<!ELEMENT
config
(
OS
,
IP
,

```
gateway
)>

<!ELEMENT
OS
 (
#PCDATA
)>

<!ELEMENT
IP
 (
#PCDATA
)>

<!ELEMENT
gateway
 (
#PCDATA
)>
```

En el archivo DTD anterior, se definen los elementos, atributos y sus relaciones esperadas. En resumen, se espera que el documento XML tenga un elemento raíz llamado equipos que contenga cero o más elementos máquina. Cada elemento máquina debe tener un atributo obligatorio nome y debe contener exactamente un elemento hardware y un elemento config. El elemento hardware debe contener un elemento tipo, un elemento fabricante, un elemento procesador, un elemento memoria, uno o más elementos disco, y un elemento opcional gravadora. Los elementos tipo, fabricante, procesador, memoria y gravadora solo deben contener texto. El elemento procesador debe tener tres atributos obligatorios marca, num_nucleos y velocidade. El elemento memoria debe tener un atributo obligatorio tecnoloxía. Cada elemento disco debe ser un elemento vacío con dos atributos obligatorios tecnoloxía y capacidade. El elemento gravadora también debe ser un elemento vacío con un atributo obligatorio tipo. El elemento config debe contener exactamente un elemento OS, un elemento IP y un elemento gateway, y se espera que cada uno de estos elementos solo contenga texto.

Para validar el XML anterior con este DTD, podemos agregar una referencia al DTD en el encabezado del XML y usar un analizador XML que admita la validación DTD. A continuación se muestra el XML actualizado con la referencia al DTD:

<?xml version=

"1.0"

encoding=

"utf-8"

?>

<!DOCTYPE

equipos

SYSTEM

"equipos.dtd"

>

<

equipos

>

<

máquina

nome

=

"PC017"

>

<

hardware
>

<
tipo
>
PC Sobremesa
</
tipo
>

<
fabricante
>
Dell
</
fabricante
>

<
procesador

marca
=
"Intel"

num_nucleos
=
"4"

velocidade
=
"3,1"
>
i7
</
procesador
>

<
memoria

tecnoloxía
=
"DDR3"
>
8
</
memoria
>

<
disco

tecnoloxía
=
"SATA"

capacidade
=
"2000"

/>

<
gravadora

tipo
=
"DVD"
/>

</
hardware
>

<
config
>

f) Lleva a cabo un backup, y después realiza su restauración (1 punto)

Siga las instrucciones a continuación para crear una copia de seguridad de una base de datos en BaseX.

Conéctese a la base de datos que desea respaldar iniciando BaseX.

Elija la pestaña "Copias de seguridad" en la vista de la base de datos.

El botón "Crear copia de seguridad" se puede utilizar para iniciar una nueva copia de seguridad de la base de datos.

Elija un nombre para el archivo de respaldo y especifique la ruta donde se almacenará.

Si desea incluir los índices en la copia de seguridad, por ejemplo, elija las opciones de copia de seguridad que desea utilizar.

Para iniciar la copia de seguridad, haga clic en "Aceptar".

Puede usar los pasos a continuación para restaurar una base de datos desde una copia de seguridad de BaseX.

Cree una base de datos vacía iniciando BaseX.

Seleccione la pestaña "Copias de seguridad" en la vista de la base de datos.

Elija el archivo de copia de seguridad que desea restaurar haciendo clic en el botón "Restaurar copia de seguridad".

Por ejemplo, si desea restaurar los índices, elija las opciones de restauración que desea utilizar.

Para restaurar la base de datos desde la copia de seguridad, haga clic en "Aceptar".

g) Declara tu propia función en XQuery (1 punto)

Obtener el nombre de todas las máquinas:

swiftCopy code

for $machine in //máquina return $machine/@nome

Obtener el tipo de todas las máquinas:

for $machine in //máquina return $machine/hardware/tipo/text()

Obtener el número de núcleos de los procesadores de todas las máquinas que tienen un procesador Intel:

for $machine in //máquina[hardware/procesador/@marca = "Intel"] return $machine/hardware/procesador[@marca = "Intel"]/@num_nucleos

Obtener la capacidad total de todos los discos de todas las máquinas:

sum(//máquina/hardware/disco/@capacidade)

Obtener el nombre y sistema operativo de la máquina que tiene la dirección IP 192.168.20.105:

for $machine in //máquina[config/IP = "192.168.20.105"] return ($machine/@nome, $machine/config/OS)

h) Lleva a cabo algún tratamiento de XML/JSON original (1 punto)

XML utilizando el lenguaje de programación Python y la biblioteca ElementTree. La idea es cargar un archivo XML, acceder a los datos que nos interesan y luego guardarlos en otro archivo XML. Digamos que queremos crear un nuevo archivo XML que contenga solo el nombre, el fabricante y el espacio en disco de cada dispositivo en el archivo original.

Primero, necesitamos instalar la biblioteca ElementTree usando pip:

pip install elementtree

A continuación, creamos un archivo Python llamado "tratamiento_xml.py" y escribimos el siguiente código:

python

Copy code

import

xml.etree.ElementTree

as

ET

Cargamos el archivo XML

tree = ET.parse(

'equipos.xml'

)

root = tree.getroot()

Creamos un nuevo archivo XML con la estructura deseada

new_root = ET.Element(

'equipos'

)

for

```
 machine
in
 root.
iter
(
'máquina'
):

# Obtenemos los datos de interés

    nome = machine.attrib[
'nome'
]
    fabricante = machine.find(
'hardware/fabricante'
).text
    capacidad =
sum
(
int
(disco.attrib[
'capacidade'
])
for
 disco
in
 machine.findall(
'hardware/disco'
))


# Creamos un nuevo elemento con los datos obtenidos
```

```
    new_machine = ET.Element(
'máquina'
, {
'nome'
: nome})
    hardware = ET.SubElement(new_machine,
'hardware'
)
    ET.SubElement(hardware,
'fabricante'
).text = fabricante
    ET.SubElement(hardware,
'disco'
, {
'capacidade'
:
str
(capacidad)})


# Añadimos el nuevo elemento al archivo

    new_root.append(new_machine)

# Guardamos el nuevo archivo XML

tree = ET.ElementTree(new_root)
tree.write(
'equipos_nuevo.xml'
, encoding=
'utf-8'
, xml_declaration=
True
```

)

Este código carga el archivo "equipos.xml", recorre todas las máquinas, obtiene el nombre, fabricante y capacidad de disco de cada una, crea un nuevo elemento XML con estos datos y lo añade al nuevo archivo "equipos_nuevo.xml". Finalmente, el nuevo archivo es guardado.