INDICE

Para buscar CP SIN EL ASTERISCO: CTRL+B

***Enlazar xsd en un sml:

<agenda xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="NOMBREXSD.XSD">

-Tienen que estar los dos archivos en la misma carpeta.

***Elemento simple: Contiene solo texto, no contiene otros elementos o atributos.

```
<xs:element name="NOMBRE" type="TIPO DE DATO"/>
```

```
//TIPOS DE DATO
***Tipos de dato:
```

.xs:string

- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Ejemplo:

//DECLARACION DE ELEMENTOS

XML:

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

XSD:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

***DEFAULT y FIXED

DEFAULT: Asigna al elemento automáticamente el valor.

```
<xs:element name="color" type="xs:string" default="red"/>
```

***FIXED:Asigna al elemento automáticamente el valor, pero no se puede asignar otro valor.

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

//ATRIBUTOS
***ATRIBUTOS:

```
<xs:attribute name="xxx" type="yyy"/>
```

Ejemplo:

XML:

```
<lastname lang="EN">Smith</lastname>
```

XSD:

```
<xs:attribute name="lang" type="xs:string"/>
```

-Los atributos DEFAULT Y FIXED funcionan igual que en un ELEMENTO SIMPLE.

***USE (REQUIRED): Si no se utiliza USE el atributo es opcional por defecto, si se usa es obligatorio.

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

//RESTRICCIONES maxInclusive Y minInclusive

***XSD RESTRICCIONES:

**RESTRICCIÓN EN VALORES(CONSTRAINT maxInclusive y minInclusive):

-La siguiente restricción define que el elemento “age” no puede ser menor a “0” o mayor de “120”.

```
<xs:element name="age">
     <xs:simpleType>
     <xs:restriction base="xs:integer">
     <xs:minInclusive value="0"/>
     <xs:maxInclusive value="120"/>
     </xs:restriction>
     </xs:simpleType>
</xs:element>
```

//RESTRICCION DE VALORES LIMITADOS
**RESTRICCIÓN DE VALORES LIMITADOS:

-La siguiente restricción (CONTRAINT enumeration) no permite que hayan otros valores aparte de: Audi, Gold o BMW.

-Primera opción:La restricción es propia de el elemento “car”.

```
<xs:element name="car">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:enumeration value="Audi"/>
               <xs:enumeration value="Golf"/>
               <xs:enumeration value="BMW"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-Segunda opción: En este caso la restricción “carType” se puede usar en otros elementos ya que no pertenece a “car”.

```
<xs:elementname="car" type="carType"/>

<xs:simpleType name="carType">
     <xs:restriction base="xs:string">
          <xs:enumeration value="Audi"/>
          <xs:enumeration value="Golf"/>
          <xs:enumeration value="BMW"/>
     </xs:restriction>
</xs:simpleType>
```

//RESTRICCION POR SECUENCIA DE VALORES
**RESTRICCIÓN EN UNA SERIE DE VALORES:

*(CONSTRAINT pattertn): Define secuencias de caracteres.

-El siguiente ejemplo limita el contenido del elemento con solo un carácter que puede estar en minúscula y entre la a y la z incluidos [a-z].

```
<xs:element name="letter">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="[a-z]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo solo deja introducir 3 caracteres entre la A y Z incluidos, pero en mayúscula.

```
<xs:element name="initials">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="[A-Z][A-Z][A-Z]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo solo deja introducir x o y o z.

```
<xs:element name="choice">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="[xyz]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo limita el contenido a 3 letra las cuales pueden estar entre la a y la z incluidos, tanto mayúsculas como minúsculas.

```
<xs:element name="initials">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo limita el contenido en 5 dígitos los cuales pueden estar entre el 0 y 9 incluidos.

```
<xs:element name="prodid">
     <xs:simpleType>
          <xs:restriction base="xs:integer">
               <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

//RESTRICCION POR SECUENCIA DE VALORES CON OPERADORES “* , ”+” y “|”
***OTRAS RESTRICCIONES EN SERIES DE VALORES:

-El siguiente ejemplo acepta el 0 o mas ocurrencias de minúsculas que deben estar entre la a y la z incluidas. “*”

```
<xs:element name="letter">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="([a-z])*"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo acepta 1 o mas pares de letras, siendo una minuscula y otra mayuscula, por ejemplo, sToP y hOlAlA validaria pero STOP, sTOP o hOlAla no validaria.”+”

```
<xs:element name="letter">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="([a-z][A-Z])+"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo usa OR para aceptar un valor u otro, en este caso el genero, que son “male” o “female”.

```
<xs:element name="gender">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="male|female"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

//RESTRICCION CON CANTIDAD EXACTA DE NUMEROS “{x}”

-El siguiente ejemplo permitirá exactamente 8 caracteres, números entre el 0 y 9 incluidos y letras tanto mayúsculas como minúsculas entre la a y la z incluidos.

```
<xs:element name="password">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:pattern value="[a-zA-Z0-9]{8}"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

//RESTRICCION PARA RESPETAR ESPACIOS EN BLANCO

***RESTRICCIÓN DE ESPACIOS EN BLANCO(whiteSpace):

-El siguiente ejemplo no se remplazaran los espacios en blanco del contenido del elemento.

```
<xs:element name="address">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:whiteSpace value="preserve"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo remplazara todos los espacios en blanco,tab,enter con espacios en blanco.

```
<xs:element name="address">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:whiteSpace value="replace"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

//RESTRICCION POR CANTIDAD DE CARACTERES CON length

***RESTRICCIONES CON (CONSTRAINT LENGTH)

-El siguiente ejemplo validara el contenido de un elemento con exactamente 8 caracteres.

```
<xs:element name="password">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:length value="8"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

-El siguiente ejemplo validara el contenido de un elemento con un mínimo de 5 caracteres y un máximo de 8.

```
<xs:element name="password">
     <xs:simpleType>
          <xs:restriction base="xs:string">
               <xs:minLength value="5"/>
               <xs:maxLength value="8"/>
          </xs:restriction>
     </xs:simpleType>
</xs:element>
```

***COMPEX ELEMENT: Elemento XML que contiene otros elementos y/o atributos.

-Cuatro tipos de Complex Elements:

-Elementos vacíos.
-Elementos que contienen solo otros elementos.
-Elementos que contienen solo texto.
-Elementos que contienen los dos tanto elementos como texto.
-Cada uno de estos elementos pueden también pueden contener atributos.

EJEMPLOS:
//ELEMENTO VACIO CON complexType + atributo
-El siguiente es un elemento vacio.

XML:

```
<product id="1345"/>
```

XSD:

```
<xs:element name="product">
     <xs:complexType>
          //Hacemos que el atributo sea un numero positivo
          <xs:attribute name="prodid" type="xs:positiveInteger"/>
     </xs:complexType>
</xs:element>

OR

<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
     <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>

//ELEMENTOS DENTRO DE UN ELEMENTO complexType, SOLO PARA ESE ELEMENTO
```

XML:

```
<employee>
	<firstname>John</firstname>
	<lastname>Smith</lastname>
</employee>
```

XSD:

-Opción 1:Solo el elemento employee usara este complex type, es importante que aparezcan en el mismo orden que en el XML.

```
//Declaramos el elemento "employee"
<xs:element name="employee">
	//Elementos que contiene mas elementos.
	<xs:complexType>
		//sequence hace que dentro del elemento hayan una
		secuencia de elementos.
		<xs:sequence>
			<xs:element name="firstname" type="xs:string"/>
			<xs:element name="lastname" type="xs:string"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
//ELEMENTOS DENTRO DE UN ELEMENTO complextype, PARA ESE ELEMENTO Y OTROS
```

-Opción 2: En este ejemplo "employee" tiene el atributo type hace refierencia al complexType "personinfo".

```
<xs:element name="employee" type="personinfo"/>

<xs:complexType name="personinfo">
	<xs:sequence>
		<xs:element name="firstname" type="xs:string"/>
		<xs:element name="lastname" type="xs:string"/>
	</xs:sequence>
</xs:complexType>
```

-De esta forma varios elementos pueden usar este complexType.

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

<xs:complexType name="personinfo">
	<xs:sequence>
		<xs:element name="firstname" type="xs:string"/>
		<xs:element name="lastname" type="xs:string"/>
	</xs:sequence>
</xs:complexType>
```

//EXTENSION DE LOS ELEMENTOS DE UN complexType haciendo referencia a otro

complexType.
-Opción 3: El siguiente ejemplo te permite extender la lista de los elementos de un complextype.

```
<xs:element name="employee" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
     <xs:sequence>
          <xs:element name="firstname" type="xs:string"/>
          <xs:element name="lastname" type="xs:string"/>
     </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
     //complexContent y extension permiten añadir elementos a un
       complexType.
     <xs:complexContent>
          //El atributo base hace referencia al complexType
          “personinfo”.
          <xs:extension base="personinfo">
               <xs:sequence>
                    <xs:element name="address" type="xs:string"/>
                    <xs:element name="city" type="xs:string"/>
                    <xs:element name="country" type="xs:string"/>
               </xs:sequence>
          </xs:extension>
     </xs:complexContent>
</xs:complexType>
```

XML:

```
<food type="dessert">Ice cream</food>
```

XSD:

XML:

```
<description>
     It happened on <date lang="norwegian">03.03.99</date>....
</description>
```

XSD: