ZPFFVX

Anonymkod/Anonymous code: ZPFFVX

TENTAMEN/EXAMINATION

SDXML VT2024

Modeller och språk för hantering av

Tentamen/Written exam 3,5 hp/hec

IB166N (SU) XMLT AF

Torsdag/Thursday 2024-05-30
09:00-13:00

Poäng Points	Betyg Grade
29%	F

Markera besvarade frågor med 'X' / Mark answered questions with 'X'													Antal blad # sheets
1	2	3	4	5	6	7	8	9	10	11	12		
X	X	X	X	X	X								13

Vakt kontrollerat antal blad: GM.

Obs! Denna sida måste ligga överst - This page should be placed in front
Avlägsna tomma blad före inlämningen
Remove empty sheets before handing in the exam
Fyll i samtliga uppgifter på sidhuvudet på varje blad
Please fill in all information in the header on each sheet

Stockholms universitet

Tentamen/Examination: SDXML

Uppgift nr/Question number: 1

Datum/Date: 2024-05-30

Blad nr/Page number: 1

Anonymkod / Anonymous code: ZPFFVX

1) A well formed XML is one that is syntactically correct, has all matching opening and closing tags. 0,5

2) PSVI is Post schema validation infoset. 0,5

3) Processing-instruction is a type of command used in xquery. 0

4) JSON schema is a set of rules a JSON object must follow, to be considered valid. 1-

5)

6) SVG - scalable vector graphics. It is used to render animations, shapes, circles on a web browser. 0,5

```
{ "name" : "abcd",
  "colors" : [ "blue", "green", "yellow" ],
  "sizes" : [ {"code" : "abc", "price" : 1000 } ],
  "available" : true
}
```

One more size

~~[redacted]~~

~~Park and TreeType can be the root element.~~

Park is better for root element.

XML document with sample data.

~~<Park name="abc" size=100>~~

<?xml version="1.0" encoding="UTF-8"?>

<Park name="abc" size=100>

<TreshCan volume=100 longitude=10 latitude=20>

</TreshCan>

<TreeType name="banyan" maximumHeight=100 averageLifeInYears=200>

<TreeTypeInPark number=100 planted=10:03:20>

</TreeTypeInPark>

</TreeType>

</Park>

more Park?
more Tree types?
tree types in no park?

XML schema - <? xml version = "1.0" encoding = "UTF-8

<schema>
missing in previous page
<element name = db type = dbType > </ element >

< complexType name = dbType >
sequence?
< element name = Park type = ParkType >
</ element > </ complexType >

< complexType name = ParkType >
< attribute name= [redacted] type = String > </ attribute >
< attribute name= size type = Integer > </ attribute >
sequence? multiplicities?
< element name = TrashCan type = TrashCanType >
</ element >
< element name = TreeType type = TreeTypeType >
</element >
</ complexType >

< complexType name = TrashCanType >
< attribute name = volume type = Integer ></attribute >
< attribute name = longitude type = Number ></attribute
</schema>

```
<attribute name = latitude type = Number > </attribute>
</complexType>

<complexType name = TreeTypeType >

< attribute name = name type = String> </attribute>

< attribute name = maximumHeight Type = Integer >
  </ attribute >
< attribute name = averageLifeInYears type = Integer >
  </attribute >

< element name = TreeTypeInPark type = TreeTypeInPark
  Type■ minOccurs = 1 maxOccurs = unbounded >
</element>
</complexType>
< complexType name = TreeTypeInParkType >
  < attribute name = number type = Integer>
    </attribute >
< attribute name = planted type = Date >
    </attribute >
</ complexType >
</schema>
```

1,5

a) element Genres {

for $g in [redacted] distinct-values(// Genre)

let $M := for $a in //Movie [redacted] where ~~exists~~ (some $a/Genre = $g[redacted])

return element Movie { attribute [redacted]

Title { [redacted] $a/[redacted]@title[redacted] }, attribute

NumberOfOtherGenres [redacted] { count($a/Genre) - 1 } }

return element Genre { attribute Name { $g[redacted] /text()[redacted] } }

literal

, $M }

1,5

b) element Result {

for $g in distinct-values(//genre)

let $sm := for $s in . // showing

where $s/../../text() = $g

return element showing

{ attribute starttime { $s/@start

time }, attribute movie { $s/../../

/@title }, attribute cinema { $s/../@name },

attribute hall { $s/@hall } }

return element genre { attribute name { $g/text },

$sm }

}

c) <D>

<Movie Title>

0

1

</D>

```
<xsl:transform >
<xsl:output method="html">
<xsl:template match="/" >
[redacted]
<h1> Cinemas </h1>
[redacted] <table border: 1 >
[redacted]
<tr>
<th> Cinema </th>
<th> Number of halls </th>
<th> Number of showings </th>
<th> Number of different movies shown</th>
</tr>
[redacted]<xsl: for-each select = "//Cinema" >
[redacted]
<xsl:element name[redacted]="@Name" >
</xsl:element>
```

html, body ?

duplicates

tr, td?

Stockholms universitet

Tentamen/Examination: SDXML

Uppgift nr/Question number: 5

Datum/Date: 2024-05-30

Blad nr/Page number: 7

Anonymkod / Anonymous code: ZPFFVX

<xsl:value-of

a) SELECT XMLELEMENT(NAME "Bank",
XMLATTRIBUTE(NAME "Code", code),
XMLATTRIBUTE(NAME "Name", name),
XMLATTRIBUTE(NAME "HighestRateEver",
XMLQUERY(max(Rate))

FROM Bank, AccountType

GROUP BY Bank

join-condition?

No XML in result.

b) SELECT XMLELEMENT(NAME "Result",
XMLAGG? XMLELEMENT(NAME "Bank",
XMLATTRIBUTE(NAME "Name", name),
XMLATTRIBUTE(NAME "Code", code))

[illegible] FROM [illegible]

FROM Bank, AccountType

join?

freeWithdrawals?

c)

```
SELECT XMLELEMENT
( NAME "Result",
XMLAGG ( SELECT XMLELEMENT (
    NAME "AccountType", XMLATTRIBUTES
( [redacted] NAME "bankcode" AS code),

XMLATTRIBUTE ( NAME "bankname" AS name),
XMLATTRIBUTE ( NAME "name" AS name),

FROM Bank, AccountType
WHERE AccountType[rates] > 4 AND
```

join condition

?

0