

I10-002 braindumps

XML Master XML Master

I10-002: XML Master: Professional V2

Practice Exam: I10-002 Exams

Exam Number/Code: I10-002

Exam Name: XML Master: Professional V2

Questions and Answers: 80 Q&As

([XML Master](#))



Exam : [I10-002](#)

"XML Master: Professional V2", also known as I10-002 exam, is a XML Master certification. With the complete collection of exam questions, Just4Study has assembled to take you through 80 Q&As to your I10-002 exam preparation. In the I10-002 exam resources, you will cover every field and category in XML Master Certification helping to ready you for your successful XML Master Certification.

The exam questions cover the latest real test and with all the correct answer. we promise the Q&A for XML Master XML Master I10-002 (XML Master: Professional V2) examination of original title complete coverage. I10-002 exam questions help you pass the exam.

Just4Study I10-002 Feature:

* High quality - High quality and valued for the I10-002 Exam: 100% Guarantee to Pass Your I10-002 exam and get your XML Master certification.

* Authoritative - Authoritative braindumps with complete details about I10-002 exam.

* Cheaper - Our Just4Study products are cheaper than any other website. With our completed XML Master resources, you will minimize your **XML Master XML Master** cost and be ready to pass your I10-002 exam on Your First Try, 100% Money Back Guarantee included!

* Free - Try free XML Master demo before you decide to buy it in <http://www.Just4Study.com>.

Just4Study Guarantee:

Just4Study provides the most competitive quality of all exams for the customers, we guarantee your success at the first attempt with only our Certification Question&Answers, if you do not pass the I10-002 exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

Free I10-002 Demo Download

Just4Study offers free demo for XML Master I10-002 exam (XML Master: Professional V2). You can check out the interface, question quality and usability of our practice exams before you decide to buy it. We are the only one site can offer demo for almost all products.

The Questions & Answers cover the latest real test and with all the correct answer. we promise the Q&A for **XML Master XML Master I10-002** examination of original title complete coverage. I10-002 Questions & Answers help you pass the exam. Otherwise, we will give you a full refund.

VUE/Prometric Code: I10-002

Exam Name: XML Master: Professional V2(XML Master)

Questions and Answers: 80 Q&A

[XML Master I10-002](#) Test belongs to one of the XML Master certified test, if needs to obtain the XML Master certificate, you also need to participate in other related test, the details you may visit the [XML Master](#) certified topic, in there, you will see all related XML Master certified subject of examination.

Just4Study professional provide XML Master I10-002 the newest Q&A, completely covers I10-002 test original topic.

With our complete XML Master resources, you will minimize your XML Master cost and be ready to pass your I10-002 tests on Your First Try, 100% Money Back Guarantee included!

Just4Study Help You Pass Any IT Exam

[Just4Study.com](#) offers incredible career enhancing opportunities. We are a team of IT professionals that focus on providing our customers with the most up to date material for any IT certification exam. This material is so effective that we Guarantee you will pass the exam or your money back.

Exam : XML Master Certification

I10-002

Title : XML Master: Professional V2

1. Push the Exhibit Button to load the referenced "XML Document".

Select which of the following correctly describes the output results of an XSLT transformation of the "XML Document" using the "XSLT Style Sheet".

Note that the XSLT processor can output transformation results as a document. Line feeds and indents are not reflected.

A. <record>

```
<data>100</data>
```

```
</record>
```

B. <record xmlns="urn:xmlmaster:test">

```
<data>100</data>
```

```
</record>
```

C. <record xmlns="urn:xmlmaster:test">

```
<data xmlns= " " >100</data>
```

```
</record>
```

D. <record>

```
<data>100</data>
```

```
<data>70</data>
```

```
</record>
```

E. <record xmlns="urn:xmlmaster:test">

```
<data>100</data>
```

```
<data>70</data>
```

```
</record>
```

F. <record xmlns="urn:xmlmaster:test">

```
<data xmlns= " " >100</data>
```

```
<data xmlns= " " >70</data>
```

```
</record>
```

Answer: F

2. Push the Exhibit Button to load the referenced "XML Document".

Assume that the character "3" is obtained from the "XML document". Select which XSLT style sheet correctly performs the transformation. (Multiple answers possible. Select two.)

A. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```
<xsl:template match= " / " >
```

```
<xsl:apply-templates select= " //data[x='1'][y='2'] " />
```

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

</xsl:stylesheet>

B. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " //data[(attribute::x='1') and (text()='3')] " />

</xsl:template>

</xsl:stylesheet>

C. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " //data[self='3'] " />

</xsl:template>

</xsl:stylesheet>

D. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " //data[self::*='3'] " />

</xsl:template>

</xsl:stylesheet>

Answer: BD

3. Push the Exhibit Button to load the referenced "XML Document".

[XML Document]

```
<root><data>Imnop</data></root>
```

Assume that the "XML document" is changed to the "Results XML Document." Select which XSLT style sheet correctly performs the transformation.

Note that the XSLT processor can output transformation results as a document.

[Results XML Document]

```
<Imnop/>
```

Or

```
<Imnop></Imnop>
```

A. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " root/data " />

</xsl:template>

<xsl:template match= " data " >

<xsl:element name="<xsl:value-of select='!'/>" />

</xsl:template>

</xsl:stylesheet>

B. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " root/data " />

</xsl:template>

<xsl:template match= " data " >

<xsl:element name="{ . }"/>

</xsl:template>

</xsl:stylesheet>

C. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match= " / " >

<xsl:apply-templates select= " root/data " />

</xsl:template>

<xsl:template match= " data " >

<xsl:element name="."/>

</xsl:template>

</xsl:stylesheet>

D. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```

<xsl:template match= " / " >
<xsl:apply-templates select= " root/data " />
</xsl:template>
<xsl:template match= " data " >
<xsl:text disable-output-escaping="no"><</xsl:text>
<xsl:value-of select="."/>
<xsl:text disable-output-escaping="no"/></xsl:text>
</xsl:template>
</xsl:stylesheet>

```

Answer: B

4. Push the Exhibit Button to load the referenced "XML document".

[XML Document]

```
<root><data>Imnop</data></root>
```

Assume that the "XML Document" is changed to the "Results XML Document." Select which XSLT style sheet correctly performs the transformation.

Note that the XSLT processor can output transformation results as a document.

[Results XML Document]

```
<ZZZ><YYY>Imnop</YYY></ZZZ>
```

A. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```
<xsl:include href="exam.xsl" />
```

```
<xsl:template match= " / " >
```

```
<xsl:apply-templates select= " root " />
```

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

```
<xsl:template match= " root " >
```

```
<AAA><BBB><xsl:value-of select= " data " /></BBB></AAA>
```

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

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

[exam.xsl]

```
<xsl:stylesheet version= " 1.0 " xmlns:xsl= " http://www.w3.org/1999/XSL/Transform " >
```

```
<xsl:template match= " //root " >
```

```
<ZZZ><YYY><xsl:value-of select= " data " /></YYY></ZZZ>
```

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

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

B. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```
<xsl:import href="exam.xsl" />
```

```
<xsl:template match= " / " >
```

```
<xsl:apply-templates select= " root " />
```

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

```
<xsl:template match= " root " >
```

```
<AAA><BBB><xsl:value-of select= " data " /></BBB></AAA>
```

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

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

[exam.xsl]

```
<xsl:stylesheet version= " 1.0 " xmlns:xsl= " http://www.w3.org/1999/XSL/Transform " >
```

```
<xsl:template match= " //root " >
```

```
<ZZZ><YYY><xsl:value-of select= " data " /></YYY></ZZZ>
```

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

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

C. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```
<xsl:include href="exam.xsl" />
```

```
<xsl:template match= " / " >
```

```
<xsl:apply-templates select= " root " />
```

```

</xsl:template>
<xsl:template match= " root " >
<AAA><BBB><xsl:value-of select= " data " /></BBB></AAA>
</xsl:template>
</xsl:stylesheet>

```

[exam.xml]

```

<xsl:stylesheet version= " 1.0 " xmlns:xsl= " http://www.w3.org/1999/XSL/Transform " >
<xsl:template match= " root " >
<ZZZ><YYY><xsl:value-of select= " data " /></YYY></ZZZ>
</xsl:template>
</xsl:stylesheet>

```

D. <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

```

<xsl:import href="exam.xml" />
<xsl:template match= " / " >
<xsl:apply-templates select= " root " />
</xsl:template>
<xsl:template match= " root " >
<AAA><BBB><xsl:value-of select= " data " /></BBB></AAA>
</xsl:template>
</xsl:stylesheet>

```

[exam.xml]

```

<xsl:stylesheet version= " 1.0 " xmlns:xsl= " http://www.w3.org/1999/XSL/Transform " >
<xsl:template match= " root " >
<ZZZ><YYY><xsl:value-of select= " data " /></YYY></ZZZ>
</xsl:template>
</xsl:stylesheet>

```

Answer: A

5. Select which of the following correctly describes the results of performing a validation check on "XML Document". Assume that the XML parser correctly processes the XML schema noNamespaceSchemaLocation attribute and the schemaLocation attribute.

- A. Valid
- B. The coding for the XML Schema Document is not appropriate; therefore, an error is thrown (initial error) when processing the "testml.xsd" import element
- C. The coding for the XML Schema Document is not appropriate; therefore, an error is thrown (initial error) when processing the "testml.xsd" "<xs:element ref="rec:record" maxOccurs="unbounded" />"
- D. No processing error, but is not valid.

Answer: A

6. What must you write in XSLT style sheet (1) to process the following "XML Document" and obtain the following "transform results"? Select the correct answer below. Note that "#" indicates a line feed, and "=" indicates a tab. Assume that the XSLT processor can output transformation results as a document.

- A. Nothing needs to be written.
- B. <xml:space="preserve"/>
- C. <xsl:preserve-space elements="content"/>
- D. <xsl:strip-space elements="doc body"/>

Answer: D

7. Push the Exhibit Button to load the referenced "XML Document 1" and "XML Document 2," and process XML using "DOM Processing."

Select which of the following is the most appropriate expression of the results under XML 1.0. Line feeds and/or indents are not reflected in the results.

- A. <root2 xmlns="urn:xmlmaster:EX2">

```
<data xmlns= " urn:xmlmaster:EX1 " >string value</data>
```

```
</root2>
```

B. <root2 xmlns="urn:xmlmaster:EX2">

```
<data>string value</data>
```

```
</root2>
```

C. <root2 xmlns="urn:xmlmaster:EX2">

```
<data xmlns= " urn:xmlmaster:EX1 " />
```

```
</root2>
```

D. <root2 xmlns="urn:xmlmaster:EX2">

```
<data/>
```

```
</root2>
```

Answer: A

8. Push the Exhibit Button to load the referenced "XML Document". Choose the XML Schema Document that correctly defines the structure of "XML Document".

A. <xs:schema

```
xmlns:xs= " http://www.w3.org/2001/XMLSchema "
```

```
targetNamespace= " urn:xmlmaster:testml "
```

```
xmlns:tns= " urn:xmlmaster:testml " >
```

```
<xs:element name= " TestML " type= " tns:testmlType " />
```

```
<xs:complexType name= " testmlType " >
```

```
<xs:sequence>
```

```
<xs:element ref= " tns:record " maxOccurs= " unbounded " />
```

```
</xs:sequence>
```

```
</xs:complexType>
```

```
<xs:element name= " record " type= " tns:recordType " />
```

```
<xs:complexType name= " recordType " >
```

```
< xs:attribute name="level" type= " xs:int " />
```

```
< xs:attribute name="data" type= " xs:int " />
```

```
</xs:complexType>
```

```
</xs:schema>
```

B. <xs:schema

```
xmlns:xs= " http://www.w3.org/2001/XMLSchema "
```

```
targetNamespace= " urn:xmlmaster:testml "
```

```
xmlns:tns= " urn:xmlmaster:testml " >
```

```
<xs:element name= " TestML " type= " tns:testmlType " />
```

```
<xs:complexType name= " tns:testmlType " >
```

```
<xs:sequence>
```

```
<xs:element ref= " tns:record " maxOccurs= " unbounded " />
```

```
</xs:sequence>
```

```
</xs:complexType>
```

```
<xs:element name= " record " type= " tns:recordType " />
```

```
<xs:complexType name= " tns:recordType " >
```

```
< xs:attribute ref="tns:level" />
```

```
< xs:attribute ref="tns:data" />
```

```
</xs:complexType>
```

```
< xs:attribute name="tns:level" type= " xs:int " />
```

```
< xs:attribute name="tns:data" type= " xs:int " />
```

```
</xs:schema>
```

C. <xs:schema

```
xmlns:xs= " http://www.w3.org/2001/XMLSchema "
```

```
targetNamespace= " urn:xmlmaster:testml "
```

```
xmlns:tns= " urn:xmlmaster:testml " >
```

```

<xs:element name= " TestML " type= " tns:testmlType " />
<xs:complexType name= " testmlType " >
<xs:sequence>
<xs:element ref= " tns:record " maxOccurs= " unbounded " />
</xs:sequence>
</xs:complexType>
<xs:element name= " record " type= " tns:recordType " />
<xs:complexType name= " recordType " >
< xs:attribute ref="tns:level" />
< xs:attribute ref="tns:data" />
</xs:complexType>
< xs:attribute name="level" type= " xs:int " />
< xs:attribute name="data" type= " xs:int " />
</xs:schema>

```

D. <xs:schema

```

xmlns:xs= " http://www.w3.org/2001/XMLSchema "
targetNamespace= " urn:xmlmaster:testml "
xmlns:tns= " urn:xmlmaster:testml " >
<xs:element name= " TestML " >
<xs:complexType>
<xs:sequence>
<xs:element name= " record " maxOccurs= " unbounded " >
<xs:complexType>
<xs:attribute name= " tns:level " type= " xs:int " />
<xs:attribute name= " tns:data " type= " xs:int " />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

```

Answer: C

9. Push the Exhibit Button to load the referenced "XML Document".

[XML Document]

```

<TestML xmlns="urn:xmlmaster:testml">
<record level="1" data="100" />
<record level="2" data="250" />
</TestML>

```

Choose the XML Schema Document that does not correctly define the structure of the "XML Document".

A. <xs:schema

```

xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:xmlmaster:testml"
xmlns:tns="urn:xmlmaster:testml" >
<xs:element name="TestML" type=" tns:testmlType " />
<xs:complexType name="testmlType">
<xs:sequence>
<xs:element ref=" tns:record " maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:element name="record" type=" tns:recordType " />
<xs:complexType name="recordType">
<xs:attribute name="level" type="xs:int" />

```

```
<xs:attribute name="data" type="xs:int" />
</xs:complexType>
</xs:schema>
```

B. <xs:schema

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:xmlmaster:testml"
xmlns="urn:xmlmaster:testml" >
<xs:element name="TestML" type="testmlType" />
<xs:complexType name="testmlType">
<xs:sequence>
<xs:element ref="record" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:element name="record" type="recordType" />
<xs:complexType name="recordType">
<xs:attribute name="level" type="xs:int" />
<xs:attribute name="data" type="xs:int" />
</xs:complexType>
</xs:schema>
```

C. <xs:schema

```
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:xmlmaster:testml" >
<xs:element name="TestML" type="testmlType" />
<xs:complexType name="testmlType">
<xs:sequence>
<xs:element ref="record" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:element name="record" type="recordType" />
<xs:complexType name="recordType">
<xs:attribute name="level" type="xs:int" />
<xs:attribute name="data" type="xs:int" />
</xs:complexType>
</xs:schema>
```

D. <schema

```
xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:xmlmaster:testml"
xmlns:tns="urn:xmlmaster:testml">
<element name="TestML" type="tns:testmlType" />
<complexType name="testmlType">
<sequence>
<element ref="tns:record" maxOccurs="unbounded" />
</sequence>
</complexType>
<element name="record" type="tns:recordType" />
<complexType name="recordType">
<attribute name="level" type="int" />
<attribute name="data" type="int" />
</complexType>
</schema>
```

Answer: C

10. Which of the following describes the most correct call order of the ContentHandler interface methods when parsing

the following "XML Document" using a non-validating SAX parser? This question reflects line feeds within the XML document.

[XML Document]

<a>

c

A. startDocument - startElement - characters - startElement - characters - characters - characters - endElement - characters - endElement - endDocument

B. startDocument - startElement - ignorableWhitespace - startElement - ignorableWhitespace - characters - ignorableWhitespace - endElement - ignorableWhitespace - endElement - endDocument

C. startDocument - startElement - startElement - characters - endElement - endElement - endDocument

D. startDocument - startElement - startElement - characters - characters - endElement - endElement - endDocumentW

Answer: A

11. Which of the following correctly describes the DOM (Level 2) Node interface?

A. The Node interface can be used to change the value (nodeValue) of the DOM element node (Element)

B. The Node interface can be used to change the name (nodeName) of the DOM element node (Element)

C. The Node interface can be used to change the value (nodeValue) of the DOM attribute node (Attr)

D. The Node interface can be used to change the name (nodeName) of the DOM attribute node (Attr)

Answer: C

12. Select which of the following correctly describes WSDL. (WSDL 1)

A. WSDL assumes SOAP as the message transmission form

B. When WSDL is defined by a combination of style="rpc" and use="encoded", then encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" must be designated

C. When WSDL is defined by a combination of style="rpc" and use="encoded", then the encodingStyle attribute cannot be designated

D. WSDL may be defined by a combination of style="rpc" and use="literal"

Answer: D

13. Select which of the following DOM (Level 2) nodes does not hold a value (returns "null"). (Multiple answers possible. Select two.)

A. Attr

B. Comment

C. Element

D. Document

Answer: CD

14. Push the Exhibit Button to load the referenced "testml.xsd".

Assume that "testml.xsd" is defined. Without rewriting this XML Schema Document ("testml.xsd"), create a new, separate XML Schema Document to partially change the schema definition to write a cellPhone element as a child element of the person element. As a result, the following "XML Document" will be valid against the new schema.

Which of the following correctly describes the new XML Schema Document?

Assume the XML parser correctly processes the XML schema schemaLocation attribute.

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

<xs:import schemaLocation="testml.xsd" />

<xs:complexType name="personType">

<xs:sequence>

<xs:element ref=" name " />

<xs:element ref=" phone " />

```

<xs:element ref= " cellPhone " />
</xs:sequence>
</xs:complexType>
<xs:element name= " cellPhone " type= " xs:string " />
</xs:schema>

```

```

B. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:include schemaLocation="testml.xsd" />
<xs:complexType name="newPersonType" substitutionGroup="personType">
<xs:sequence>
<xs:element ref= " name " />
<xs:element ref= " phone " />
<xs:element ref= " cellPhone " />
</xs:sequence>
</xs:complexType>
<xs:element name= " cellPhone " type= " xs:string " />
</xs:schema>

```

```

C. <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:redefine schemaLocation= " testml.xsd " >
<xs:complexType name= " personType " >
<xs:complexContent>
<xs:extension base= " personType " >
<xs:sequence>
<xs:element ref= " cellPhone " />
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:redefine>
<xs:element name= " cellPhone " type= " xs:string " />
</xs:schema>

```

D. It is not possible to implement a function of the type proposed.

Answer: C

15. Push the Exhibit Button to load the referenced "XML Document".

Create an XML Schema Document for "XML Document". The definitions of this XML Schema Document require that the value of the level attribute of the record element must be singularly unique within the XML document, and further, that the level attribute of the scenario element must reference the value of the level attribute of the record element.

Select which of the following correctly describes what should be written in "XML Schema" document (1).

```

A. <xs:element name="TestML" type="testmlType">
< xs:unique name= " LEVEL " >
<xs:selector xpath= " record " />
<xs:field xpath= " record/@level " />
</ xs:unique >
< xs:ref name= " levelRef " refer= " LEVEL " >
<xs:selector xpath= " scenario " />
<xs:field xpath= " scenario/@level " />
</ xs:ref >
</xs:element>

```

```

B. <xs:element name="TestML" type="testmlType">
< xs:unique name= " LEVEL " >
<xs:selector xpath= " record " />
<xs:field xpath= " @level " />
</ xs:unique >

```

```
< xs:ref name= " levelRef " refer= " LEVEL " >
<xs:selector xpath= " scenario " />
<xs:field xpath= " @level " />
</ xs:ref >
</xs:element>
C. <xs:element name="TestML" type="testmlType">
< xs:key name= " LEVEL " >
<xs:selector xpath= " record " />
<xs:field xpath= " record/@level " />
</ xs:key >
< xs:keyref name= " levelRef " refer= " LEVEL " >
<xs:selector xpath= " scenario " />
<xs:field xpath= " scenario/@level " />
</ xs:keyref >
</xs:element>
D. <xs:element name="TestML" type="testmlType">
< xs:key name= " LEVEL " >
<xs:selector xpath= " record " />
<xs:field xpath= " @level " />
</ xs:key >
< xs:keyref name= " levelRef " refer= " LEVEL " >
<xs:selector xpath= " scenario " />
<xs:field xpath= " @level " />
</ xs:keyref >
</xs:element>
Answer: D
```

[I10-002 Braindumps](#)

Related I10-002 Exams

[I10-001](#) XML Master Basic V2

[I10-002](#) XML Master: Professional V2

Other XML Master Exams

[I10-002](#) [I10-001](#)