# MMDP 234 XML/XSL

# Final Exam Preparation Guide

Final Exam will be a “hands on”, “closed book” test.

* You can use notes (digital, hand-written)
* You can use your previous works as a reference.
* You can also use the Internet. Before the test, choose the website you are planning to use for reference. There is a link to such a website (w3schools) posted on the Lessons tab < Resources folder.

**Restrictions:** You will not be allowed to

1. use books of any kind during the test.
2. use your laptop during the test.
3. use e-mail.
4. flash drives are not allowed in the computers during the test. Prepare the material you will be using during the test and place it in a separate folder on your flash drive. You will copy that folder to your workstation before the test starts.

The test covers material from **Tutorials 4, 6, and 7** of our text book. The test will look like a tutorial with the steps to follow written out.

Following is the list of topics we have learned. Most of these topics will appear on the test. You can use this list as a start for your cheat sheet.

### Tutorial 4: Schema

1. Starting a schema file, <xs:schema <xmlns:xs="http://www.w3.org/2001/XMLSchema"> tag
2. Different schema designs: salami slice, Russian doll, Venetian Blind.
3. Simple types:
   1. Element with textual content

<xs:element name=”LName” type=”xs:string />

b. Attribute

<xs:attribute name=”patId” type=”xs:string />

* 1. Derived restricted data type:
     1. enumeration, minExclusive - maxExclusive, pattern (regular expressions)

<xs:simpleType name="stageType">

<xs:restriction base="xs:string">

<xs:enumeration value="I" />

<xs:enumeration value="II" />

</xs:restriction>

</xs:simpleType>

1. Complex types:
   1. An element containing attributes and some textual content

<xs:complexType name="perfComplex">

<xs:simpleContent>

<xs:extension base="perfType">

<xs:attribute name="scale" type="scaleType" use="required" />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

* 1. An element containing child elements (and maybe attributes)

<xs:complexType name=”Name”>

<xs:sequence>

<xs:element name=”first” type=”xs:string />

<xs:element name=”last” type=”xs:string />

</xs:sequence>

<xs:attribute name=”ID” type=”xs:string />

</xs:complexType>

1. Referencing elements

<xs:element name="patient">

<xs:complexType>

<xs:sequence>

<xs:element ref="lastName"/>

1. Creating element and attribute groups and referencing them later.
2. Using XML standard built-in data types (string, decimal, ID etc.)
3. Applying a schema on an XML document

<patients xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance> xsi:noNamespaceSchemaLocation="pschema.xsd">

1. Using Visual Studio to validate XML document against the schema

**“Placing schema into a Namespace” topic will not be present on the test.**

### Tutorial 6 Intro to XSL and XPath

1. Attaching an XSL style sheet to the XML document
2. XPath -- referencing nodes of the document tree
   1. Moving through the node tree using a path: patients/patient/name
   2. moving one level up ../
   3. Referencing attributes: patients/patient/**@id**
   4. Referencing current element -- select=” **.”**
3. Starting a stylesheet:

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

1. The root template: <xsl:template match="/">
2. Literal result elements <html>, <body> etc.
3. Declaring output method
4. Extracting element textual content (value) or attribute value

<xsl:value-of select="portfolio/date" />

<xsl:value-of select="@id" />

1. Creating template that matches some node.

<xsl:template match="day">

1. Applying template <xsl:apply-templates select="portfolio/stock/day" />
2. Inserting attribute values -- using {} instead of <xsl:value-of…>

<a href="{../link}">

1. Sorting node sets

<xsl:apply-templates select="portfolio/stock[category='Industrials']">

<xsl:sort select="sName" />

</xsl:apply-templates>

1. Using choose element
2. Using if element
3. Using comparison operators and functions

<xsl:when test="@current &lt; @open">

<img src="down.gif" />

</xsl:when>

1. XPath predicates: stock[position()>1] or stock[position()=2]

### Tutorial 7 XSL and XPath for Computations and Numbering

1. Numbering nodes and formatting the numbers

<xsl:number value="position()" format="1)" />

1. XPath functions:
   1. count() <xsl:value-of select="count(//customer)" />
   2. sum() <xsl:value-of select="sum(//@qty)" />
   3. Using mathematical operations: <xsl:value-of select="@qty \* @price" />
   4. Formatting resulting number: <xsl:value-of select="format-number(@qty \* @price, '#,##0.00')" />
   5. Adding white space: &#160;
   6. Declaring and referencing variables:

<xsl:variable name="group"

select="customers/customer [starts-with(@CID,$filter)]" />

Referencing:

<xsl:apply-templates select="$group" />

**Material from pp. 404 - 432 will not be present on the test**