Announcements

- In-class Student Presentations
  - Presentations last for the duration of a class (i.e., 50 min).
  - Must include two components (each would take roughly half of the time):
    - Presentation (Google slides or any other online support that allows for easy collaboration: share the link with me)
    - A coding demo (similar to what we do in class: your slides should include tasks, you can share code with the class if you want them to work on it, you must share the solution)
  - Timeline
    - Discuss the outline of the presentation with me before you start preparing (this should be the first slide of your presentation)
    - A draft of the presentation slides and coding demo (to be shared before and after class) must be submitted to me **at least 1 week prior to the presentation** to allow for feedback and iterations.
  - Note: Everyone is responsible to attend and know topics covered during those presentations for the exams and assignments.

Extensible Markup Language (XML)

Cpt S 321

Washington State University

Motivation

- Just about all applications need to load/save files
- Over the years – data stored in inflexible binary formats
- Not easily extended when you want to add a new property to some object
- Not human-readable and not able to modify in simple text editors
- XML aims to fix that

What it is

- It’s a markup language
- From Wikipedia: “A (document) markup language is a modern system for annotating a document in a way that is syntactically distinguishable from the text.”
- Similar to HTML (hyper text markup language) but instead of a list of possible tags you can create whatever you need in an XML document

What it does

- Stores descriptive information in a file in a structured way
- General form is:
- <tag_name>content</tag_name>
- Where content can consist of additional nested tags

Example 1: List of Classes

```
<schedule>
  <class>Cpt S 317</class>
  <class>Cpt S 322</class>
  <class>Cpt S 360</class>
</schedule>
```

Example 2: Personnel List

```
<faculty>
  <professor salary="12345" tenured="no">Bob Smith</professor>
  <professor salary="54321" tenured="yes">Wendy Smith</professor>
</faculty>
```

Attributes

Example 3: Product List

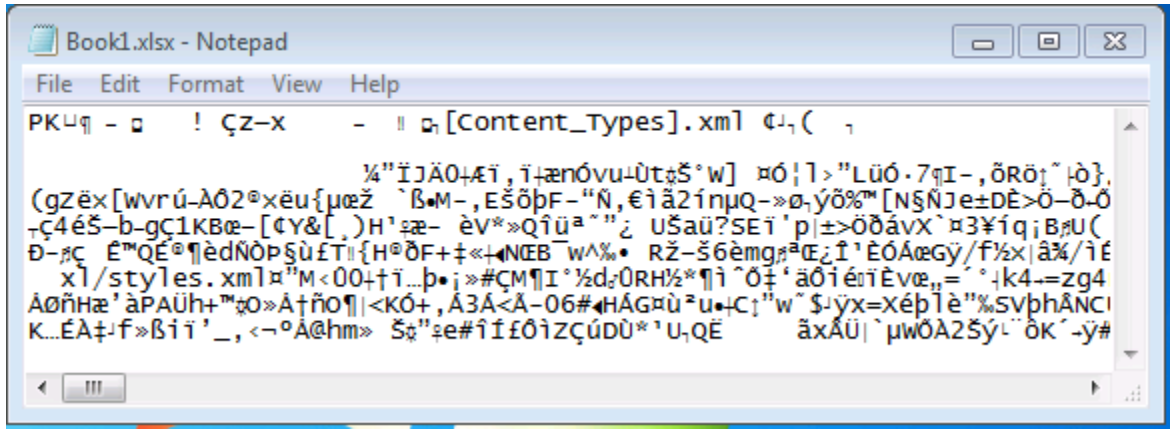
```
<all_products>
  <product>
    <name>Some product name 1</name>
    <inventory_count>100</inventory_count>
    <id>12345</id>
  </product>
  <product>
    <name>Some product name 2</name>
    <inventory_count>50</inventory_count>
    <id>54321</id>
  </product>
</all_products>
```

What do all these have in common?

- Not much other than they’re all valid XML
- That’s the point – XML elements and content can be whatever you want
- XML is used in many modern file formats, including Microsoft Excel xlsx files

But when I look at an .xlsx file in a text editor…

- It looks like a bunch of binary gibberish
- That’s because it is a binary file, it’s a zip file
- But inside are XML documents



Reading/Writing XML Files in C#

- .NET has a few XML reading/writing capabilities
- XmlReader / XmlWriter: High-performance, forward-only cursors for reading or writing an XML stream
- XmlConvert: A lot of useful methods that will help with conversion
- XmlDocument: Represents an XML document in a W3C-style DOM (Document Object Model)
- XDocument: LINQ to XML DOM
- Why so many?
- Why might you want the XmlReader/XmlWriter when the XmlDocument class can load and save XML documents?

Building a File Format to Save Spreadsheet Data

- You may have seen this coming: your spreadsheet application should be able to load and save files (HW9)
- We will use XML documents
- Specification for Excel file format is hundreds of pages and you'd need to decompress the .zip
  - Probably not reasonable to have you implement loading / saving of that format
  - But there are .NET libraries that handle the .zip files and to make a minimal loader / saver would be doable if you're motivated to investigate in your spare time
- Let's discuss just using a single .XML file

Need a root node

<my_xml_document>
  <what_goes_in_here>?</ what_goes_in_here>
</my_xml_document>

XML Terminology: Element

- Content below is from World Wide Web Consortium (W3C) documentation:
- What is an XML Element?
- An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.
- An element can contain:
  - other elements
  - text
  - attributes
  - or a mix of all of the above...
- (source: http://www.w3schools.com/xml/xml_elements.asp)

XML Terminology: Attribute

- Attributes are stored in the start tag of an element before it closes with the > character.
- <xmlElementTagName **attr1=“Hello” attr2=“world”**>Element content</xmlElementTagName>

Example

```
<faculty>
  <professor salary="12345" tenured="no">Bob Smith</professor>
  <professor salary="54321" tenured="yes">Wendy Smith</professor>
</faculty>
```

In **bolded red** is a professor element (the first of two).

Example

```
<faculty>
  <professor salary="12345" tenured="no">Bob Smith</professor>
  <professor salary="54321" tenured="yes">Wendy Smith</professor>
</faculty>
```

In **bolded red** is the salary attribute for the first professor element.

In **bolded blue** is the tenured attribute for the first professor element.

Example

```
<faculty>
  <professor salary=“12345” tenured=“no”>Bob Smith</professor>
  <professor salary=“54321” tenured=“yes”>Wendy Smith</professor>
</faculty>
```

The **faculty** element is everything in this example.

.NET Framework Terminology

- .NET is mostly aligned with the terminology mentioned on the past few slides
- XmlNode is the base class for XmlElement, XmlAttribute and XmlDocument
- The XmlDocument can store an entire document with many nested XmlElements and these elements can have XmlAttributes within them, yet all three of these classes inherit from XmlNode
- Will need to make heavy use of the documentation while you're coding

Links for additional reading

- http://www.w3schools.com/xml/xml_whatis.asp
- System.Xml Namespace
- XmlDocument class
- XmlElement class
- XmlNode class
- XDocument class

XML Saving

- Now we can think about various options for a design that allows us to save spreadsheet information to a file
- Option 1
  - Spreadsheet class has a save method that pulls properties out of the cells as needed and writes all the XML itself
  - The Cell class wouldn't have any loading or saving-specific code
- Option 2
  - Implement IXmlSerializable in the Cell class so that it has the ability to write itself to XML and load its properties from XML as well
  - Locks you in to using XmlWriter and XmlReader

XML Loading

- Cases are analogous to the saving code
- Option 1
  - Spreadsheet class has a load method that pulls properties out of the files as needed and sets them in the cells
  - The Cell class wouldn't have any loading or loading-specific code
- Option 2
  - Implement IXmlSerializable in the Cell class so that it has the ability to load itself from XML (and save for the saving case)
  - Locks you in to using XmlWriter and XmlReader

Example Cell XML Element

```
<cell name="B2">
  <text>=A1+A2</text>
  <bgcolor>#FF80AA</bgcolor>
</cell>
```

- Could add extra properties in future versions without breaking backwards compatibility
- Older versions would just ignore the unknown elements and load the known ones

Another Example Cell XML Element

<cell name="B2" text="=A1+A2" bgcolor="#FF80AA" />

- Has the same information as the cell on the previous slide
- This raises the question: which is better, the one above or the one below?

```
<cell name="B2">
  <text>=A1+A2</text>
  <bgcolor>#FF80AA</bgcolor>
</cell>
```

Coming up with the XML format

- The XML format is arbitrary, so you decide how to write the elements
- General tip: anything that might be extended or stored in a different form in the future should be a child element as opposed to an attribute
- Example scenario: suppose an early version of your app supports only one type of border on cells and whether or not the border is present is a Boolean property. Think about extending it in the future to support line thickness, line pattern (dashed vs. solid), sides of the cell with visible or hidden borders and so on.

Coming up with the XML format

<cell border="shown">…</cell>

vs.

```
<cell>
  <border>
    <shown>true</shown>
  </border>
</cell>
```