

# טכנולוגיית XML

ושימושיה בהנדסת תוכנה

---

אוהד ברזילי



פיתוח מערכות תוכנה מבוססות Java

# תקציר

---

- XML והטכנולוגיות הנלוות לה מרכזות עניין ופעילות בעולם התוכנה באקדמיה ובתעשייה
- על אף שהשפה עצמה פשוטה מאוד (ואולי בגלל זאת) להפיכתה לתקן נפוץ יש **השלכות מהפכניות** על ייצוג המידע, התקשורת, האינטגרציה והתצורה של מערכות תוכנה מודרניות
- ל- XML יש מקום מרכזי בהפיכת ה- Web Services לכלי שימושי
- בשיעור היום נסקור כמה מהטכנולוגיות מבוססות XML ונעמוד על משמעותן בהנדסת תוכנה

# נושאי מחקר ופיתוח

---

□ פיתוח, תקנון ומימוש כלי XML נמצא בחזית המחקר האקדמי והפעילות בתעשייה

□ חלק מהתקנים שיוזכרו כאן הם מהשנה האחרונה (!)

□ לחלקם אין מימושים יעילים או מימושים כלל

□ תקנים רבים ממתינים לביצוע מחקר תיאורטי שיוכיח את נכונותם

# XML

---

eXtensible Markup Language

XML היא שפת תגיות (בדומה ל HTML למשל)

בשונה מ HTML התגיות בה לא מוגדרות מראש

השפה תוקננה ע"י W3C

היא תת שפה של SGML

Standard Generalized Markup Language

XML נועדה לתאר **מידע** העובר בין תוכניות מחשב

# יתרונות XML

---

- שפה סטנדרטית
- אינה תלויה בכלי, פלטפורמה, מסד נתונים, פרוטוקול או שפת תכנות
- פשוטה וקלה להבנה ולשימוש
- קהילת מפתחי כלים, סטנדרטים ושפות לשימוש נוח ב XML
- ניתנת להרחבה (לשפה לשימושים מסויימים)
- ספריות Java לעבודה עם XML

# שימושי XML

---

# שימושי XML – תקשורת ואינטגרציה

---

□ החלפת מידע (data) בסביבה מבוזרת (באינטרנט) בצורה שאינה תלויה שפה  
■ מידע עשוי להיות מורכב (טיפוס חדש)

□ SOAP  
■ **Simple Object Access Protocol**  
■ מהווה את הבסיס לכתיבת Web Services

□ SOA  
■ **Service Oriented Architecture**  
■ גישה חדשה לארגון המידע והתהליכים הארגוניים  
■ לכל תהליך אירגוני מנשק אחד שבו יכולים להשתמש רכיבים אחרים ביישום ו\או שותפים עסקיים מחוץ לארגון

# XML and the Internet

---

□ כאשר מידע עובר בין מחשבים, ארגונים, פלטפורמות ופרוטוקולים יש צורך אמיתי בתקן (פורמט) שכולם יוכלו "להתיישר לפיו"

□ דוגמא: Ajax

□ דוגמא אחרת: XML based Web Services



# שימושי XML - קובצי תצורה

---

- שימוש בקובצי תצורה (קונפיגורציה) מאפשר לנו לשנות מידע דינאמית (בזמן ריצה) ללא צורך לקמפל מחדש את כל האפליקציה
- לפעמים המידע הדינאמי מורכב
- התחביר של המידע עשוי להשתנות לאורך זמן
- ראינו איך בעזרת XML ניתן להגדיר תצורת יישומים מבוססי Eclipse
- שימוש ב 2-way editor
- XML שימושי מאוד בתצורת שרתי אינטרנט

# שימושי XML – הפשטת ייצוג המידע

---

□ הפרדה בין הלוגיקה (מקודדת בשפת תכנות) ובין המידע (מקודד ב-XML)

□ ניתן לטעון עצמים ומחלקות המוגדרים ב-XML לכל יישום ללא תלות בשפת התכנות שבה נכתב היישום

□ הגדרת מסדי נתונים מבוססי XML

# שימושי XML – הגדרת שפות וכלי תוכנה

---

- ❑ Ant
- ❑ XHTML
- ❑ Scalable Vector Graphics (SVG)
- ❑ Chemical Markup Language
- ❑ Mathematical Markup Language (MathML)
- ❑ Open Financial Exchange

# תכנות הצהרתי

## Declarative (Meta) Programming

---

□ בכמה מקרים מטשטשים ההבדלים:

■ בין כלי עזר לשפת תכנות

■ בין pre-processor לקומפיילר

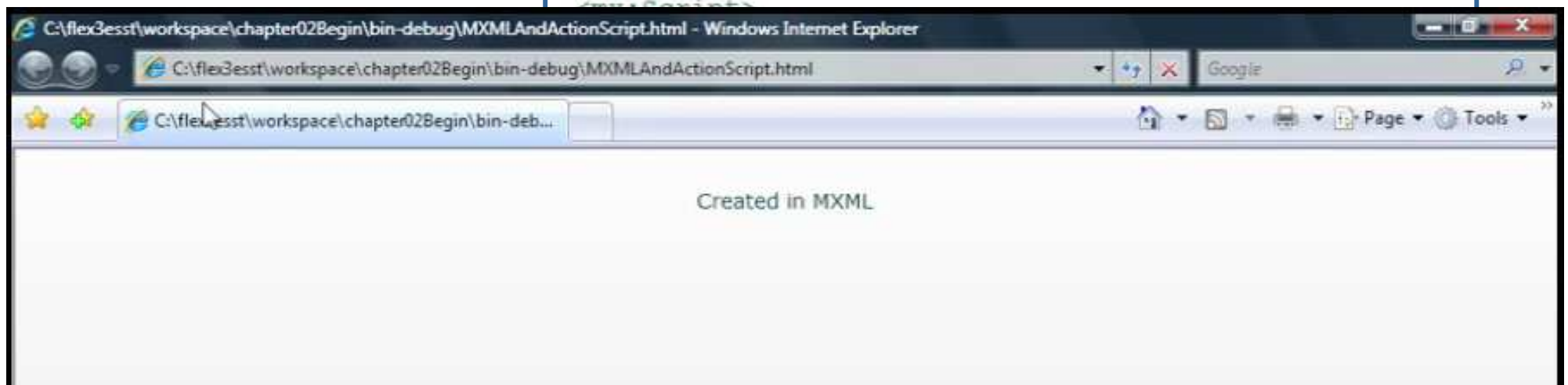
■ בין קונפיגורציה ללוגיקה

□ וכך ניתן למצוא שפות תכנות מבוססות XML

■ לדוגמא MXML של Adobe

# דוגמא: תכנות הצהרתי ב Flex3

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
3   layout="vertical" backgroundColor="#eeeeee" >
4
5   <mx:Label text="Created in MXML" fontSize="14"/>
6
7 </mx:Application>
8
```





# Crash Course on XML Technology

# XML is Based on Markup

*Markup indicates*

*structure and semantics*

```
<bibliography>
```

```
  <paper ID= "object-fusion">
```

```
    <authors>
```

```
      <author>Y. Papakonstantinou</author>
```

```
      <author>S. Abiteboul</author>
```

```
      <author>H. Garcia-Molina</author>
```

```
    </authors>
```

```
    <fullPaper source="fusion"/>
```

```
    <title>Object Fusion in Mediator Systems</title>
```

```
    <booktitle>VLDB 96</booktitle>
```

```
  </paper>
```

```
</bibliography>
```

*Decoupled from  
presentation*

# Elements and their Content

`<bibliography>`

**element name**

**Element  
Content**

`<paper ID="object-fusion">`

`<authors>`

`<author>Y. Papakonstantinou</author>`

`<author>S. Abiteboul</author>`

`<author>H. Garcia-Molina</author>`

`</authors>`

`<fullPaper source="fusion"/>`

`<title>Object Fusion in Mediator Systems</title>`

`<booktitle>VLDB 96</booktitle>`

`</paper>`

element

**Empty  
Element**

`</bibliography>`

**Character content**



# Element Attributes

```
<bibliography>
```

Attribute name

```
<paper ID="object-fusion">
```

Attribute Value

```
<authors>
```

```
<author>Y. Papakonstantinou</author>
```

```
<author>S. Abiteboul</author>
```

```
<author>H. Garcia-Molina</author>
```

```
</authors>
```

```
<fullPaper source="fusion"/>
```

```
<title>Object Fusion in Mediator Systems</title>
```

```
<booktitle>VLDB 96</booktitle>
```

```
</paper>
```

```
</bibliography>
```



# Elements and attributes

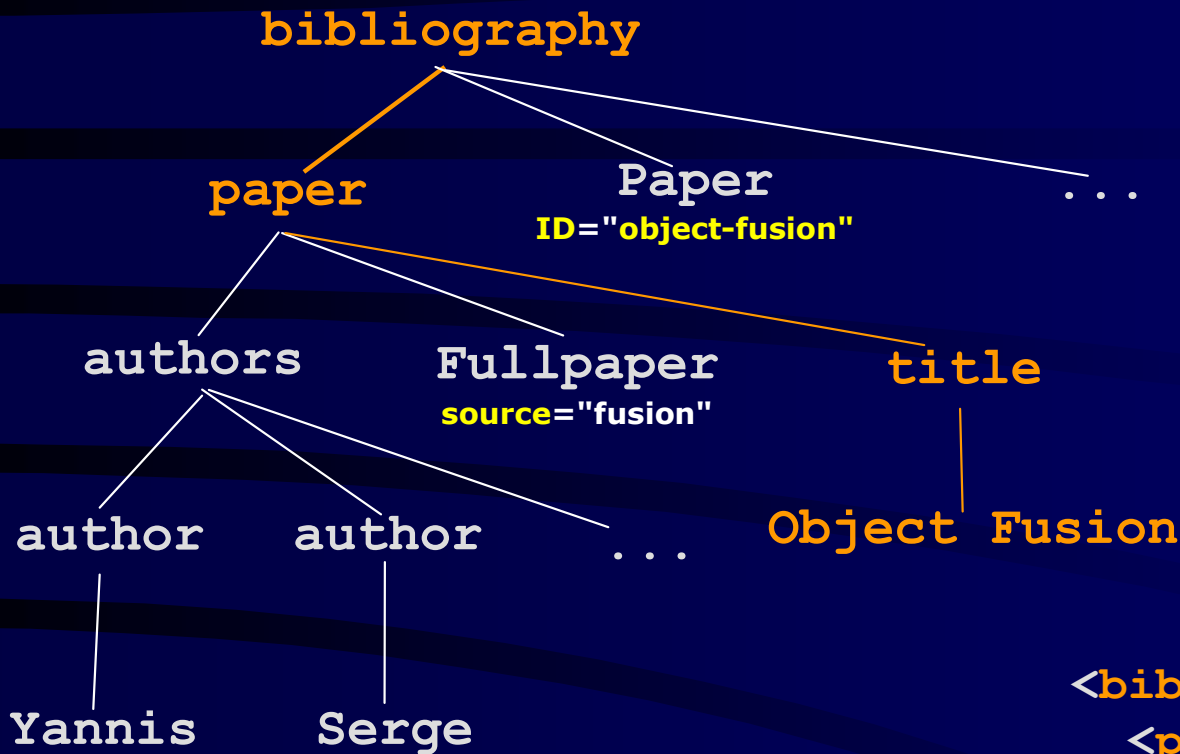
---

- Attributes and elements are somewhat interchangeable
- Example using just elements:

```
<name>  
  <first>David</first>  
  <last>Matuszek</last>  
</name>
```
- Example using attributes:

```
<name first="David" last="Matuszek"></name>
```
- You will find that elements are easier to use in your programs--this is a good reason to prefer them
- Attributes often contain metadata, such as unique IDs
- Generally speaking, browsers display only elements (values enclosed by tags), not tags and attributes

# XML = Labeled Ordered Trees



can also represent

- *relational and*
- *object-oriented data*

*≈ semistructured data*  
*≈ labeled trees/graphs*

```
<bibliography>
  <paper ...>
    <authors>
      <author>Yannis</author>
      <author>Serge</author>
      ...
    </authors>
    <title>Object Fusion</title>
  ...
</paper>
</bibliography>
```



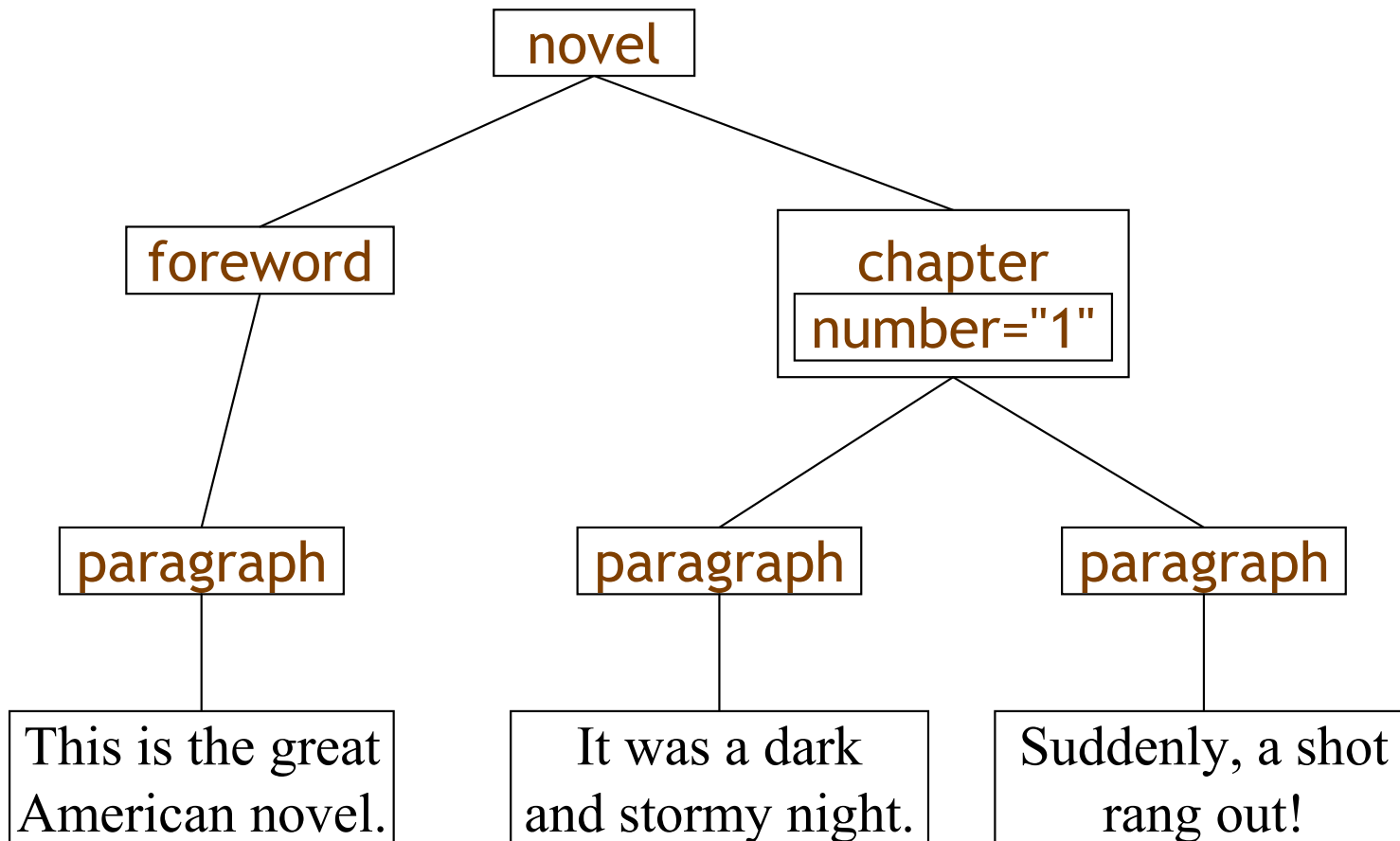
# Another well-structured example

---

```
<novel>
  <foreword>
    <paragraph> This is the great American novel.
  </paragraph>
</foreword>
<chapter number="1">
  <paragraph>It was a dark and stormy night.
</paragraph>
  <paragraph>Suddenly, a shot rang out!
</paragraph>
</chapter>
</novel>
```

# XML as a tree

An XML document represents a hierarchy; a hierarchy is a tree ■



# In Search of the Lost Structure & Semantics

How do I learn and use the **element structure** of a document?

How do I **share structure** and **metadata/semantics** with my community?

How to make all this **automatable**?



# תקינות מסמך XML

---

- ❑ **DTD** (Document Type Definition)
- ❑ **XML** Schema
- ❑ **RELAX NG** (**RE**gular **L**anguage for **X**ML **N**ext **G**eneration)

# An XML example

---

```
<novel>
  <foreword>
    <paragraph>This is the great American novel.</ paragraph>
  </foreword>
  <chapter number="1">
    <paragraph>It was a dark and stormy night.</paragraph>
    <paragraph>Suddenly, a shot rang out!</paragraph>
  </chapter>
</novel>
```

- An XML document contains (and the DTD describes):
  - Elements, such as **novel** and **paragraph**, consisting of *tags* and *content*
  - Attributes, such as **number="1"**, consisting of a *name* and a *value*
  - Entities (not used in this example)



# A DTD example

---

```
<!DOCTYPE novel [  
  <!ELEMENT novel (foreword, chapter+)>  
  <!ELEMENT foreword (paragraph+)>  
  <!ELEMENT chapter (paragraph+)>  
  <!ELEMENT paragraph (#PCDATA)>  
  <!ATTLIST chapter number CDATA #REQUIRED>  
>
```

- A **novel** consists of a **foreword** and one or more **chapters**, in that order
  - Each **chapter** must have a **number** attribute
- A **foreword** consists of one or more **paragraphs**
- A **chapter** also consists of one or more **paragraphs**
- A **paragraph** consists of parsed character data (text that cannot contain any other elements)

# An expanded DTD example

---

```
<!DOCTYPE novel [  
  <!ELEMENT novel  
    (foreword, chapter+, biography?, criticalEssay*)>  
  <!ELEMENT foreword (paragraph+)>  
  <!ELEMENT chapter (section+ | paragraph+)>  
  <!ELEMENT section (paragraph+)>  
  <!ELEMENT biography(paragraph+)>  
  <!ELEMENT criticalEssay (section+)>  
  <!ELEMENT paragraph (#PCDATA)>  
>
```

# Another example: XML

---

```
<?xml version="1.0"?>
<!DOCTYPE weatherReport SYSTEM
    "http://www.mysite.com/mydoc.dtd">
<weatherReport>
  <date>05/29/2002</date>
  <location>
    <city>Philadelphia</city>, <state>PA</state>
    <country>USA</country>
  </location>
  <temperature-range>
    <high scale="F">84</high>
    <low scale="F">51</low>
  </temperature-range>
</weatherReport>
```

# The DTD for this example

---

```
<!ELEMENT weatherReport (date, location,  
                           temperature-range)>  
<!ELEMENT date (#PCDATA)>  
<!ELEMENT location (city, state, country)>  
<!ELEMENT city (#PCDATA)>  
<!ELEMENT state (#PCDATA)>  
<!ELEMENT country (#PCDATA)>  
<!ELEMENT temperature-range  
      ((low, high) | (high, low))>  
<!ELEMENT low (#PCDATA)>  
<!ELEMENT high (#PCDATA)>  
<!ATTLIST low scale (C|F) #REQUIRED>  
<!ATTLIST high scale (C|F) #REQUIRED>
```

# סטנדרטים לכלי עבודה עם XML

---

- Parsers
  - **DOM** (Document Object Model)
  - **JDOM** (Java Document Object Model)
  - **SAX** (Simple API for XML)
  - **StAX** (Streaming API for XML)

# Java APIs for XML

---

- **JAXP** (Java API for XML Processing)
  - SAX, StAX, DOM, XSLT
  
- **JAXB** (Java Architecture for XML Binding)
  - Marshalling and unmarshalling java objects
  
- Distributed Interoperability
  - **JAXM** (Java API for XML Messaging)
  - **SAAJ** (SOAP with Attachments API for Java)
  - **JAX-RPC** (Java API for XML-Based RPC)
  - **JAXR** (Java API for XML Registries)

# What is JAXB?

---

- JAXB is Java Architecture for XML Binding
- SAX and DOM are *generic* XML parsers
  - They will parse any well-structured XML
- JAXB creates a parser that is *specific to your DTD*
  - A JAXB parser will parse only *valid* XML (as defined by your DTD)
- JAXB produces as output Java *source code* which *you* compile and add to your program
  - Your program will use the specific classes generated by JAXB
  - *Your program* can then read and write XML files
- DOM and JAXB both produce a tree in memory
  - DOM produces a *generic* tree; everything is a **Node**
  - JAXB produces a tree of *Objects* with names and attributes as described by your DTD



# A JAXB example

- The DTD: 

```
<!ELEMENT book (title, author, chapter+)>  
<!ELEMENT title (#PCDATA) >  
<!ELEMENT author (#PCDATA)>  
<!ELEMENT chapter (#PCDATA) >
```
- The schema: 

```
<xml-java-binding-schema>  
  <element name="book" type="class" root="true" />  
</xml-java-binding-schema>
```
- The results: 

```
public Book(); // constructor  
public String getTitle();  
public void setTitle(String x);  
public String getAuthor();  
public void setAuthor(String x);  
public List getChapter();  
public void deleteChapter();  
public void emptyChapter();
```

Note 1: In these slides we only show the class outline, but JAXB creates a *complete* class for you

Note 2: JAXB constructs names based on yours, with good capitalization style



# ניווט, שאילתות ועדכון

---

- ❑ XML Path (**XPath**)
- ❑ XML Query (**XQuery**)
- ❑ XML Query Language (**XQL**)
- ❑ XML Linking (**XLink**)
- ❑ XML Pointer (**XPointer**)

# שיטות להצגת מסמכי XML

---

- **CSS** was designed for styling HTML pages, and can be used to style XML pages
  
- **XSL** (Extensible Stylesheet Language)
  - **XSLT** (XSL Transformations)
  - **XPath**
  - **XSL-FO** (XSL Formatting Objects) is a replacement for CSS

# עוד מקורות

---

- XML: The Big Picture and Some Gory Details
  - <http://daks.ucdavis.edu/~ludaesch/Paper/2000-07-UCLA.ppt>
  
- CIT 597 Programming Languages & Techniques III
  - <http://www.cis.upenn.edu/~matuszek/cit597-2007/index.html>
  
- Java Passion
  - <http://www.javapassion.com/xml/index.html>