Τεχνολογίες Εφαρμογών Διαδικτύου

XML
Java Architecture for XML Binding (JAXB)

Δρ. Ι. Χαμόδρακας
Μέλος του Εργαστηριακού Διδακτικού Προσωπικού
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

* Μέρος του υλικού προέρχεται από τη Wikipedia και το site w3schools.org

XML: eXtensible Markup Language

- Η XML (αγγλ. αρκτ. από το Extensible Markup Language) είναι μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων.
- Οι χαρακτήρες που απαρτίζουν ένα κείμενο XML, αποτελούν είτε τη **σήμανση** είτε το **περιεχόμενό** του.
- Όλα τα αλφαριθμητικά που συνιστούν τη σήμανση, είτε ξεκινούν με το χαρακτήρα "<" και καταλήγουν στο χαρακτήρα ">", είτε ξεκινούν με το χαρακτήρα "&" και καταλήγουν στο χαρακτήρα ";".
- Ακολουθίες χαρακτήρων που δε συνιστούν τη σήμανση, αποτελούν το περιεχόμενο ενός κειμένου XML.

XML: eXtensible Markup Language

- **Ετικέτα:** ένα στοιχείο σήμανσης που ξεκινά με το χαρακτήρα "<" και καταλήγει στο χαρακτήρα ">".
- Υπάρχουν τρία είδη ετικέτας: *ετικέτες-αρχής*, για παράδειγμα <section>, *ετικέτες-τέλους*, για παράδειγμα </section>, και *ετικέτες-χωρίς-περιεχόμενο*, για παράδειγμα <line-break/>.
- **Στοιχείο:** Ένα λογικό απόσπασμα ενός κειμένου, που είτε ξεκινά με μία ετικέτα-αρχής και καταλήγει σε μία ετικέτα-τέλους, είτε αποτελείται μόνο από μία ετικέτα-χωρίς-περιεχόμενο.
- Οι χαρακτήρες που υπάρχουν μεταξύ μιας ετικέτας-αρχής και μιας ετικέτας-τέλους, συνιστούν το **περιεχόμενο** του στοιχείου
- Το περιεχόμενο μπορεί να περιέχει σήμανση, συμπεριλαμβανομένων και άλλων στοιχείων, που ονομάζονται **στοιχεία-παιδιά.**

XML: eXtensible Markup Language

- **Χαρακτηριστικό:** ένα στοιχείο σήμανσης που αποτελείται από ένα ζευγάρι *όνομα/τιμή*, το οποίο υπάρχει μέσα σε μία ετικέτα-αρχής ή σε μία ετικέτα-χωρίς-περιεχόμενο.
- Παράδειγμα: το στοιχείο *img* έχει δύο χαρακτηριστικά, τα *src* και *alt*: .
- Παράδειγμα πλήρους εγγράφου XML:

```
<?xml version="1.0" encoding='UTF-8'?>
<painting>
        <img src="madonna.jpg" alt='Foligno Madonna'/>
        <caption>This is Raphael's "Foligno" Madonna,
        painted in <date>1511</date> <date>1512</date>.
        </caption>
</painting>
```

DTD: Document Type Definition

- Ένα αρχείο **DTD** (ορισμός τύπου εγγράφου) ορίζει τα νόμιμα δομικά στοιχεία ενός εγγράφου XML, HTML και γενικότερα **SGML** (Standard Generalized Markup Language)
- Ορίζει τη δομή του εγγράφου και τη λίστα των νόμιμων στοιχείων και των χαρακτηριστικών τους.
- Ένα έγγραφο DTD συσχετίζεται με ένα έγγραφο XML (ή SGML) μέσω μιας δήλωσης τύπου εγγράφου (DOCTYPE)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

Παράδειγμα δήλωσης DOCTYPE

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<!-- the XHTML document body starts here-->

<html xmlns="http://www.w3.org/1999/xhtml"> ...
</html>
```

Element Types

- Τα έγγραφα DTD περιέχουν τουλάχιστον τους τύπους των στοιχείων που περιέχονται στα έγγραφά και τη λίστα των χαρακτηριστικών τους.

```
<!ELEMENT html (head, body)>
<!ELEMENT p (#PCDATA | p | ul | ol | table |
h1|h2|h3)*>
```

- Το στοιχείο html περιέχει τα στοιχεία head και body.
- Το στοιχείο p έχει περιεχόμενο #PCDATA: parsed character data, δηλαδή κείμενο που στο εσωτερικό του μπορεί να περιέχει άλλα στοιχεία (τα στοιχεία p, ul, ol, table, h1, h2 και h3).
- CDATA: unparsed character data, δηλαδή κείμενο του οποίου το περιεχόμενο δεν αναλύεται για σήμανση άλλων στοιχείων.
- * : μηδέν ή περισσότερα αντικείμενα, +: τουλάχιστον ένα, ?: μηδέν ή ένα. Αν δεν υπάρχει τίποτε από τα παραπάνω: ακριβώς ένα αντικείμενο στο περιεχόμενο του στοιχείου.
- `<!ELEMENT square EMPTY>`: στοιχείο χωρίς περιεχόμενο <square />

Attribute list

- Οι λίστες χαρακτηριστικών ορίζουν τα χαρακτηριστικά των στοιχείων. Περιέχουν: το όνομα του στοιχείου, τα ονόματα των χαρακτηριστικών, τον τύπο δεδομένων των χαρακτηριστικών (ή απαρίθμηση των επιτρεπτών τιμών), πιθανές default τιμές.

```
<!ATTLIST element-name attribute-name attribute-type
attribute-value>
<!ATTLIST img
    src CDATA #REQUIRED
    id ID #IMPLIED
    sort CDATA #FIXED "true"
    print (yes | no) "yes”>
```

ID: μοναδική τιμή

attribute-value: REQUIRED: απαραίτητο, FIXED: συγκεκριμένη τιμή που ακολουθεί, IMPLIED: προαιρετικό

XSD: XML Schema Definition

- XSD: Εναλλακτικός **πιο ισχυρός** τρόπος ορισμού της δομής εγγράφων XML μέσω της ίδιας της γλώσσας XML.

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com">
...
...
</xs:schema>
```

- Πάντοτε ένα σχήμα XML έχει ως στοιχείο-ρίζα το schema
- Το πρώτο χαρακτηριστικό `xmlns:xs` δηλώνει ότι τα στοιχεία και οι τύποι δεδομένων στο σχήμα προέρχονται από τον χώρο ονομάτων (namespace) `http://www.w3.org/2001/XMLSchema` και ότι ξεκινούν πάντοτε με xs:

XSD: XML Schema Definition

- Το επόμενο χαρακτηριστικό δηλώνει ότι τα στοιχεία που θα οριστούν στο σχήμα ανήκουν στο namespace
  `targetNamespace=http://www.w3schools.com`
- Σε ένα άλλο έγγραφο XML δηλώνεται συμμόρφωση προς ένα σχήμα XSD ως εξής:

```
<?xml version="1.0"?>

<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XSD: XML Schema Definition

- Το χαρακτηριστικό xmlns εντός ενός στοιχείου δηλώνει το namespace των στοιχείων που θα χρησιμοποιηθούν.
- Το χαρακτηριστικό xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" δηλώνει τη χρήση των στοιχείων από το αναγραφόμενο namespace που ξεκινούν από xsi:
- Το χαρακτηριστικό xsi:schemaLocation="http://www.w3schools.com note.xsd" δηλώνει το σχήμα που θα χρησιμοποιηθεί για το namespace (πρώτη τιμή είναι το namespace και δεύτερη η θέση του σχήματος)

Στοιχεία XSD

- Απλά στοιχεία: `<xs:element name="xxx" type="yyy"/>`

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, xs:time (συνήθεις τύποι)

Τα στοιχεία μπορούν να έχουν default ή fixed τιμές

```
<xs:element name="color" type="xs:string"
default="red"/>
<xs:element name="color" type="xs:string"
fixed="red"/>
```

(δεν μπορεί να οριστεί άλλη τιμή εκτός από την προκαθορισμένη)

Σύνθετα στοιχεία XSD

- **Σύνθετα στοιχεία:** στοιχεία που περιέχουν άλλα στοιχεία, στοιχεία που περιέχουν κείμενο και άλλα στοιχεία, στοιχεία που περιέχουν μόνο κείμενο, κενά στοιχεία.

```
<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

XML Schema Definition: περισσότερα

http://www.w3schools.com/schema/default.asp

Java Architecture for XML Binding

- Η Java παρέχει το API JAXB για το μετασχηματισμό αντικειμένων σε έγγραφα XML και το αντίστροφο. Θα πρέπει να χρησιμοποιηθεί κάποια υλοποίηση (π.χ. https://jaxb.java.net/)
- Υπάρχουν 2 τρόποι να χρησιμοποιηθεί αυτό το API:
  - Να δημιουργηθεί το κατάλληλο XML schema το οποίο θα χρησιμοποιηθεί για τους μετασχηματισμούς και θα συσχετισθεί με συγκεκριμένες κλάσεις
  - Να χρησιμοποιηθούν κατάλληλα annotations εντός των κλάσεων των αντικειμένων που μετασχηματίζονται. **Απλούστερη οδός!**

Παράδειγμα JAXB: Marshalling

```
<?xml version=”1.0" encoding=”UTF-8"
standalone=”yes”?>
<customer id=”100”>
<age>22</age>
<name>Sumeet</name>
</customer>
```

Παράδειγμα JAXB: Marshalling

```
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class Customer {

    String name;
    int age;
    int id;

    public String getName() {
        return name;
    }

    @XmlElement
    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }
```

Παράδειγμα JAXB: Marshalling

```
    @XmlElement
    public void setAge(int age) {
        this.age = age;
    }

    public int getId() {
        return id;
    }

    @XmlAttribute
    public void setId(int id) {
        this.id = id;
    }
}
```

Παράδειγμα JAXB: Marshalling

```
import java.io.File;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;

public class JAXBDemo {

    public static void main(String ar[])
    {
        Customer customer = new Customer();
        customer.setId(100);
        customer.setName("Sumeet");
        customer.setAge(22);
```

Παράδειγμα JAXB: Marshalling

```
        try {
            //Marshalling
            File file = new File("D:\\file.xml");
            JAXBContext jaxbContext = JAXBContext.newInstance(Customer.class);
            Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
            jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
true);
            jaxbMarshaller.marshal(customer, file);
            jaxbMarshaller.marshal(customer, System.out);
        }
        catch(Exception ex)
        {
            System.out.println(ex);
        }
    }
}
```

OUTPUT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customer id="100">
<age>22</age>
<name>Sumeet</name>
</customer>
```

Παράδειγμα JAXB: Unmarshalling

```
<?xml version="1.0"?>
<user>
    <name>Duck Ranger</name>
    <accountNumber>45</accountNumber>
    <username>duckr</username>
    <COMMAND>list</COMMAND>
    <superUser>false</superUser>
    <PROGRAMS>
        <program>
            <executable>processor.exe</executable>
            <directory>/products/processor</directory>
        </program>
        <program>
            <executable>preprocessor.exe</executable>
            <directory>/products/preprocessor</directory>
        </program>
    </PROGRAMS>
    <lastLogin>21-10-2010</lastLogin>
</user>
```

Παράδειγμα JAXB: Unmarshalling

```
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.CollapsedStringAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

//The XMLRootElement annotation tells JAXB what is the
//element local name it needs to look for.
@XmlRootElement(name="program")
public class Program {

    private String executable;
    private String directory;

    //The adapter tells JAXB to collapse white space. This is usually
    //quite good to use for Strings.
    //The XmlElement annotation tells JAXB what's the
    //actual name of the element inside <program>
    @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
    @XmlElement(name="executable")
    public String getExecutable() {
        return executable;
    }
```

Παράδειγμα JAXB: Unmarshalling

```
public void setExecutable(String executable) {
        this.executable = executable;
    }

    @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
    @XmlElement(name="directory")
    public String getDirectory() {
        return directory;
    }

    public void setDirectory(String directory) {
        this.directory = directory;
    }

    @Override
    public String toString() {
        return "Program ["
            + (executable != null ? "executable=" + executable + ", " : "")
            + (directory != null ? "directory=" + directory : "") + "]";
    }

}
```

Παράδειγμα JAXB: Unmarshalling

```
import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
//Note that the element is all up-case. The Java class does not
//have to have exactly the same name as the element. It is
//enough that you create the right mapping. JAXB automatic
//bindings will not do that for you.
@XmlRootElement(name="PROGRAMS")
public class Programs {
    private List<Program> programs;
    @XmlElement(name = "program")
    public List<Program> getPrograms() {
        if (null==programs) {
            programs = new ArrayList<Program>();
        }
        return programs;
    }
```

Παράδειγμα JAXB: Unmarshalling

```
public void setPrograms(List<Program> programs) {
        this.programs = programs;
    }

    @Override
    public String toString() {
        return "Programs [" + (programs != null ? "programs=" +
programs : "") + "]";
    }
}
```

Παράδειγμα JAXB: Unmarshalling

```
import java.util.Date;
import java.util.List;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.adapters.CollapsedStringAdapter;
import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

@XmlRootElement(name="user")
public class User {

    private String name;
    private Integer accountNumber;
    private String userName;
    private String command;
    private Boolean superUser;
    private Date lastLogin;
```

Παράδειγμα JAXB: Unmarshalling

```
private Programs programs;

@XmlJavaTypeAdapter(CollapsedStringAdapter.class)
@XmlElement(name="name")
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
@XmlElement(name="accountNumber")
public Integer getAccountNumber() {
    return accountNumber;
}
public void setAccountNumber(Integer accountNumber) {
    this.accountNumber = accountNumber;
}
```

Παράδειγμα JAXB: Unmarshalling

```
@XmlJavaTypeAdapter(CollapsedStringAdapter.class)
@XmlElement(name="username")
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}
//As you can see, the member name does not have to correspond
//to the actual element name, as long as you have the mapping
//sorted.
@XmlJavaTypeAdapter(CollapsedStringAdapter.class)
@XmlElement(name="COMMAND")
public String getCommand() {
    return command;
}
public void setCommand(String command) {
    this.command = command;
}
```

Παράδειγμα JAXB: Unmarshalling

```
@XmlElement(name="superUser")
public Boolean getSuperUser() {
    return superUser;
}
public void setSuperUser(Boolean superUser) {
    this.superUser = superUser;
}
@XmlElement(name="lastLogin")
public Date getLastLogin() {
    return lastLogin;
}
public void setLastLogin(Date lastLogin) {
    this.lastLogin = lastLogin;
}
@XmlElement(name="PROGRAMS")
public Programs getPrograms() {
    return programs;
}
```

Παράδειγμα JAXB: Unmarshalling

```
    public void setPrograms(Programs programs) {
        this.programs = programs;
    }

    @Override
    public String toString() {
        return "User ["
            + (name != null ? "name=" + name + ", " : "")
            + (accountNumber != null ? "accountNumber=" + accountNumber
            + ", " : "")
            + (userName != null ? "userName=" + userName + ", " : "")
            + (command != null ? "command=" + command + ", " : "")
            + (superUser != null ? "superUser=" + superUser + ", " : "")
            + (lastLogin != null ? "lastLogin=" + lastLogin + ", " : "")
            + (programs != null ? "programs=" + programs : "") + "]";
    }

}
```

Παράδειγμα JAXB: Unmarshalling

- Μετατροπή εγγράφου XML σε στιγμιότυπο

```
import java.io.FileReader;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

public class Unmarshal {

    public static void main(String args[]) throws Exception {

        JAXBContext context = JAXBContext.newInstance(User.class); //1
        Unmarshaller unmarshaller = context.createUnmarshaller();
        User user = (User)unmarshaller.unmarshal(new
FileReader("./test.xml")); //2
        System.out.println(user); //3
    }
}
```