W10.a web technologies

- 理解 XML 数据结构
- 学习 XML 文档模式
- 掌握查询与转换技术
- 熟悉 XML 的应用程序接口
- 了解 XML 数据存储方法及应用场景

什么是 XML?

定义

- **XML (Extensible Markup Language)**：由 W3C 定义，用于描述数据的一种标记语言。
- 来源：基于 **SGML (Standard Generalized Markup Language)** 简化而成。

特点

1. **扩展性**：用户可自定义标签。
2. **层次结构**：标签支持嵌套，适合存储和交换复杂数据。
3. **自描述性**：通过标签，数据对人类更具可读性。

示例代码：

```
xmlCopy code<university>
    <department>
        <dept_name>Comp. Sci.</dept_name>
        <building>Taylor</building>
        <budget>100000</budget>
    </department>
</university>
```

XML 的动机与优点

- **数据交换**：在网络世界中，数据交换是核心需求。XML 是新一代数据交换标准。
- **应用广泛**：从银行（如资金转账）到科学研究（如 ChemML, MathML）都有相关应用。

对比：

- 传统格式
  ：使用纯文本+行头（plain text with line headers），如电子邮件。
  - 缺点：不支持嵌套结构，没有标准类型。
- **XML**：支持嵌套结构，允许定义新标签，具有标准的语法和解析工具。

XML 数据结构

元素与标签

- **标签 (Tag)**：表示数据的逻辑部分。
- **元素 (Element)**：从 `<tag>` 开始到 `</tag>` 结束的完整数据部分。
- 嵌套规则

  ：元素必须正确嵌套，不能交错。

  - 正确示例：

    ```
    xmlCopy code<course>
        <title>Intro to CS</title>
    </course>
    ```

  - 错误示例：

    ```
    xmlCopy code<course>
        <title>Intro to CS</course>
        </title>
    ```

属性 (Attributes)

- 属性以 `name=value` 格式定义在起始标签内。
- 示例：

```
xmlCopy code<course course_id="CS-101">
    <title>Intro to Computer Science</title>
</course>
```

属性 vs 子元素

- **属性**：适用于标识符。
- **子元素**：适用于内容。
- 例如：

  ```
  xml


  Copy code
  <course course_id="CS-101"></course>
  ```

  等价于：

  ```
  xmlCopy code<course>
      <course_id>CS-101</course_id>
  </course>
  ```

XML 文档模式（Schema）

文档类型定义（DTD）

- 用途：定义 XML 文档的结构和约束。
- 语法：

```
dtdCopy code<!ELEMENT element-name (subelements-specification)>
<!ATTLIST element-name (attributes)>
```

- 示例：

```
dtdCopy code<!ELEMENT university (department)>
<!ELEMENT department (dept_name, building, budget)>
<!ELEMENT dept_name (#PCDATA)>
```

DTD 的局限性

1. 无数据类型支持（所有值均为字符串）。
2. 对引用（ID 和 IDREF）的约束较弱。
3. 无法表达子元素的无序集合。

XML Schema

- 用途：解决 DTD 的局限性，支持类型约束、复杂对象及继承。
- 语法：基于 XML 格式，定义更为标准，但更复杂。
- 示例：

```
xmlCopy code<xs:element name="department">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="dept_name" type="xs:string"/>
            <xs:element name="budget" type="xs:decimal"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

查询与转换 XML 数据

XPath

- **作用**：用于定位 XML 文档的部分数据。
- 语法：

```
xpath


Copy code
/university/department[@dept_name='Comp. Sci.']
```

- 常用功能：
    1. `/`：根节点。
    2. `//`：任意子节点。
    3. `@`：访问属性。

XSLT

- **作用**：用于从 XML 转换为另一种 XML 或 HTML。
- 使用规则（template）结合 XPath 定义转换逻辑。

XQuery

- **作用**：查询 XML 数据，语法类似 SQL。
- 示例：

```
xqueryCopy codefor $x in /university/course[credits > 3]
return <course_id>{ $x/@course_id }</course_id>
```

XML 数据存储

存储方式

1. **非关系存储**：
    - 平面文件：简单但缺乏并发支持。
    - 专用 XML 数据库（如 eXist-db）：提供查询优化。
2. **关系存储**：
    - 将 XML 转换为关系模型，支持现有数据库技术。
    - 示例：
        - **字符串表示法**：整个 XML 存储为字符串。
        - **树形表示法**：每个节点存储为关系表的一行。

应用接口

1. **SAX (Simple API for XML)**：事件驱动的解析接口。
2. **DOM (Document Object Model)**：基于树的解析和操作接口。

b. web technologies

1. 理解语义网（Semantic Web）的概念和目的。
2. 掌握资源描述框架（RDF）的基本结构。
3. 学习本体（Ontology）和知识图谱（Knowledge Graph）。
4. 了解链接数据（Linked Data）的原则及其应用。
5. 学习 SPARQL 查询语言，用于语义网数据的操作和查询。

语义网（Semantic Web）

定义

语义网是对当前 Web 的扩展，其目的是让信息具有明确的意义，便于计算机和人类更好地协作。

- 引用：“语义网使信息具有明确的意义，从而更好地支持计算机和人类的协作。”——Tim Berners-Lee

语义网的目标

1. 构建支持**数据网络（Web of Data）**的技术栈。
2. 通过语义化数据，使计算机能够更有效地完成工作。
3. 支持可信的网络交互。

为什么需要语义网?

在当今网络环境中，许多任务需要整合不同来源的数据，如：

- **企业数据整合**：整合酒店预订、交通安排、会议信息等分散数据。
- **科学数据挖掘**：整合生物化学、遗传学、药物学和患者数据库的数据。
- **数字图书馆**：跨多个格式和语言的文献交叉引用。

挑战：人类可以理解数据的关联，但机器无法直接处理不同语言、格式或语法的数据。
解决方案：语义网提供标准化技术，促进数据整合与语义处理。

资源描述框架（RDF）

定义

RDF 是一种用于在 Web 上表示信息的框架，核心是**三元组（Triples）**的结构：

- **Subject**（主语）：表示实体。
- **Predicate**（谓语）：表示关系。
- **Object**（宾语）：表示目标。

RDF 的特点

1. 支持跨模式的数据合并，即使底层模式不同。
2. 通过三元组形式形成 RDF 图，每个三元组对应一个“节点-弧-节点”的链接结构。

示例：

- `(Person, hasPet, Cat)`
- `(University, locatedIn, City)`

RDF 的直观形式就是一种“关系网络”，便于表达实体间的关联。

RDFS 与 OWL

RDFS（Resource Description Framework Schema）

- 提供描述 RDF 词汇表的基本能力，例如定义类和属性的层次结构。

OWL（Web Ontology Language）

- 在 RDFS 之上提供更强大的知识表示能力，是一种面向复杂知识表达的语义网语言。
- 特点：
    1. 基于计算逻辑的语言。
    2. 能够验证知识的一致性。
    3. 推导隐含知识，使隐式知识显式化。

OWL 的作用

- OWL 文档通常被称为“本体（Ontology）”，它定义了**概念、关系及其约束**，便于机器理解和推理。

Ontology

定义

- 对某一领域的**显式概念化**的规范描述。
- 数据与ontology的区别：
    - **数据**：表示具体的原子事实。
    - **本体**：表示知识，可以通过逻辑表达存在性和全局性的语句。

ontology与知识库

- 在语义网中，“本体”和“知识库”有时可以互换使用，但二者仍有微妙的区别：
    - 本体更多地关注概念和关系的定义。
    - 知识库则包括具体的事实数据。

链接数据（Linked Data）

定义

Linked Data 是一组通过标准化方式互联的数据集，支持语义网工具的访问和操作。

- **目标**：跨 Web 构建分布式数据。
- **应用**：可以轻松访问或转换现有数据库（如关系型数据库、XML 数据库等）。

设计原则

1. **使用 URI**：将实体标识为唯一的 URI。
2. **使用 HTTP URI**：便于通过 Web 访问这些 URI。
3. **提供有用的信息**：返回 RDF 格式的标准化数据。
4. **包含链接**：链接到其他 URI，便于发现更多内容。

链接数据云图：展示全球范围内链接数据的增长趋势，从 2007 年到 2023 年，涵盖了从政府数据到社交网络等多种应用领域。

SPARQL：查询语义网数据

定义

SPARQL（SPARQL Protocol and RDF Query Language）是由 W3C 开发的标准化查询语言，用于操作 RDF 数据。

功能

1. 支持对 RDF 数据的图模式匹配。
2. 允许操作多个数据源，无论数据是否本地存储为 RDF。
3. 查询结果可以是结果集或 RDF 图。

查询结构

SPARQL 的查询基于**模式（Patterns）**匹配：

- 模式描述的是一种三元组结构，其中可以包含变量，用于匹配实际数据中的关系。

示例

1. 查询所有国家：

```
sparqlCopy codeSELECT ?country
WHERE {
    ?country rdf:type dbo:Country
} LIMIT 100
```

2. 查询出生地包含“United Kingdom”的艺术家：

```
sparqlCopy codePREFIX dbprop: <http://dbpedia.org/property/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

SELECT ?artist ?place
WHERE {
    ?artist rdf:type dbpedia-owl:Artist.
    ?artist dbprop:birthPlace ?place.
    ?place rdf:type dbpedia-owl:Country.
    ?place rdfs:label ?label.
    FILTER regex(?label, "^United Kingdom").
}
```

查询结果

SPARQL 引擎返回所有匹配模式的资源。用户可以通过公开的 SPARQL 端点（如 DBPedia）访问开放的语义网数据。

知识图谱（Knowledge Graph）

定义

知识图谱通过语义网技术构建复杂实体和关系的网络。

- 应用领域：搜索引擎（如 Google 知识图谱）、推荐系统、智能助理等。

特点

- 将语义网中定义的本体与实际数据结合起来，提供强大的信息检索和推理能力。