Multimedia Informationssysteme 2
VO/KU SS 2005 - 706.054, 706.055
Denis Helic

Introduction

Course

- Multimedia Informationssysteme 2 (VO 706.054)
- Multimedia Informationssysteme 2 (KU 706.055)
  - Elective (optional) course for Telematics Master
  - Subject area catalogue: Information Systems, Multimedia

Lecturer

- Who is talking?
  - Name: Denis Helic
  - How can I reach him?
    - Office: IICM, Inffeldgasse 16c, 1st Floor, Room D.2.05
    - Office hours: Wednesday from 11 til 12
    - Phone: +43-316/873-5617
    - email: dhelic@iicm.edu

Language

- Lectures in English
- Communication in German/English
- If in German: please informally (Du)!
- Constructional Example: German/English
- Examination: German/English

Organization of the course

- Lectures: Thursday, 11:15 – 12:45, HS i12
- Registration for Lectures (VO), Exercise (KU) (in TU Online System until 31.03.2006)
- Lecture Slides, Links to external resources (http://coronet.iicm.edu/lectures/mmis2)
  - The Web site will be updated a couple of times during the term
  - Newsgroup tu-graz.lv.mmis2 (news.tu-graz.ac.at)

Goals of the course

- In MMIS2 we deal with the Web as an application platform
  - Goal: To learn and understand the specifics of Web as an application platform
    - HTTP, URL, Client/Server, Data Formats, ...
  - Goal: To learn about the technological trends
    - What technology is currently used? (XML, (X)HTML, Web App Frameworks, Web Services, ...)
  - Goal: To learn about the trends in development methods
    - What are the software engineering methods and how to develop Web apps?
  - Goal: To learn about the software design trends
    - What are common problems and common design patterns in Web applications?
How we will achieve these goals?(1/2)

The theoretical background and overview of the current trends
Lectures, slides, links to articles, ...
Practical implementation of a Web application
Following a software (Web) engineering method
Using some of the technologies that we have discussed
Applying Web-oriented design patterns

How we will achieve these goals?(2/2)

Presentation of the achieved results
All students participate in the presentation
Discussion of the results
With the lecturer
With the peers

Topics of the course(1/3)

Data Formats: Markup Languages (XML)
Document-oriented XML
XML Publishing, XML-based Content Management Systems
Using XSLT and XSL-FO for presentation
Data-oriented XML
Data types
Storing XML data, Native XML Databases
XML as a means for Declarative Web development

Topics of the course(2/3)

Traditional Page-oriented Web applications
Server-side programming with Java (Ruby)
Design Patterns
Model-View-Controller Pattern
Web Application frameworks: Apache Struts, Ruby on Rails
Data Management in Web applications
Object/relational mappings
Frameworks and patterns: Hibernate, Data Access Object, ...

Topics of the course(3/3)

Service oriented Web applications
Web Services
SOAP, WSDL
REST architectural style
Web Services Frameworks: Apache Axis
Rich Web Clients and Web Services
AJAX

Course Calendar(1/2)

09.03. Web Eng. Intro/Intro on frameworks for practical work
16.03. XML Publishing, XSL-FO, Cocoon
23.03. XML Data Management, Native XML Databases, XQuery
30.03. Model-View-Controller (Struts)
06.04. Data Management in Web Applications (O/R Mapping, Hibernate)
04.05. Model-View-Controller (Ruby on Rails)

Course Calendar(2/2)

11.05. Web services (SOAP, WSDL)
18.05. Web services (REST, AJAX)
01.06. Project presentations (XML Pub.)
08.06. Project presentations (MVC)
22.06. Project presentations (Web Serv.)

Excercise (KU) - Organization of the course(1/7)

No lectures for KU
Registration: Required (in TU Online System)
Registration deadline: 31.03.2006
Goals:
- Implementing a Web application
- Learning about different technologies, methods
- Working in groups
- To try out something new

Prerequisites
- (X)HTML, CSS
- XML
- Java

Group work: groups of 3-5 people
Group registration with a short description
http://coronet.iicm.edu/denis/students/mmis/group.txt
Project plan (Time estimation, group members responsibilities, ...)
http://coronet.iicm.edu/denis/students/mmis/plan.txt

Documentation of Code and Design!
http://coronet.iicm.edu/denis/students/mmis/doc.pdf
KU example points to MMIS 1 (documents are same, the content is different)
Documentation is not equal to the source code
Documentation is a description of the system
- Architecture, Installation, User Guide, etc.

Presentation
- 15min + 5min discussion (if not so many groups: 20min+5min)
- Argue about your development desicions
- Why did you take MVC architecture?
- Point advantages, disadvantages

Deadlines:
- Group building and preliminary description of the project (what we are going to do): 31.03.2006
- Project plan: 04.05.2006
- Completed project: one day before presentation
- XML Pub.: 31.05.2006
- MVC: 07.06.2006
- Web Ser.: 21.06.2006

Grading:
- Grading of the KU:
  - Project plan: 25%
  - Documentation: 25%
  - Implementation: 50%
  - Project submission via e-mail: Denis Helic (dhelic@iicm.edu)

- Grading of the lecture (Oral exam):
  - Presentation 50%
  - Answering questions 50%
  - Additional points if you participate in discussion
  - E.g. you can improve your grade in this way
Important for the exam!!!
Show how you applied design patterns
Show how you applied a method
Argue why you did something!!!
Advantages/disadvantages!
Note that this is very similar to Sem/Proj or Master Exam

If you do not make a VO/KU combination
Only KU
If you good at answering questions you can improve your grade
Only VO
Oral exam, with questions going more into details
Special exam dates in TUGOnline

Exercise Topics (1/3)
Students can implement whatever Web application they like!
- Web shops
- Content management systems
- Collaborative systems
- Digital libraries

Exercise Topics (2/3)
Three Java Frameworks + One Ruby Framework
- Cocoon for XML publishing http://cocoon.apache.org/
- Struts for MVC Web applications http://struts.apache.org/
- Rails for MVC Web applications http://www.rubyonrails.com/
- Axis for Web services http://ws.apache.org/axis/

Exercise Topics (3/3)
In some cases something else can be done
You need to discuss this with me first!
- Native XML databases
- Hibernate
- REST + AJAX

Introduction to Web Engineering
Denis Helic

Web Applications - Historical Background (1/3)
The Web started as a static information system
- Web servers served static HTML documents
- Users wanted more interaction with the Web
  - Interactive Web-based phone book
  - Introduction of HTML forms
    - Need for a dynamic response from Web servers
  - Server-side programming as an answer: CGI, Perl, PHP, ...(MMIS1)

Web Applications - Historical Background (2/3)
The Web and the Internet became ubiquitous
- Faster than any other technological development in history
- Manufacturing, travel, banking, education, government
  - All are Web-enabled
- The Web is an application platform!
Web Applications - Historical Background

- E-commerce crossing national boundaries
- Porting of legacy systems to the Web
- Wireless applications → mobile Web applications
- We increasingly depend on a range of Web applications

Web Engineering - Definition

- We rely on Web applications → they need to be reliable and perform well
- To build such applications we need a sound methodology
- Process, tools, guidelines
- Web engineering
  - Systematic, scientific, engineering and management approach
  - Develop, deploy and maintain qualitative Web applications

Web Engineering - Multidisciplinary Process

- Web software engineering is similar to traditional software engineering
  - The process follows a similar methodology
  - But at each step we need to take care about special issues related to the Web
- Requirements engineering
- System analysis, system architecture and design
- Implementation
- Testing

Web apps UR vs. traditional apps UR

- New dimension of dealing with users
- In traditional SE you know your users
  - Easy to derive user requirements
- In WE you don’t know your users
  - Potentially, the whole world!

- Even if you have a target group it can change in the future
- Additionally, you need to compete with others over the users
  - Usability, accessibility, graphic design become very important
- Popularity is important!
  - Google based its search engine on popularity → a large success

Web apps SA vs. traditional apps SA

- Generic architecture of user-oriented software applications
- Process (application) logic
- Data management
- User interface

- A typical example
- Managing student records in university
- Process logic: add a student, delete a student, insert grades, exams, ...
- Data management: tables of students, grades, exams (relational databases)
- User interface: GUI with menus, buttons, text fields, etc.
Generic architecture of Web applications

User interface in Web applications

- Traditionally, developers deal mainly with process logic and data management
- User interface lately in the process and in a hurry
- Poor quality of interfaces → users complain
- The Web is appealing because of its consistent user interface
  - HTML, platform independence
  - One of the reasons for the success!
  - It is a huge advantage for developers!

Hypertext engineering in Web applications (1/2)

- Hyperlinks are a specific component of a Web-based user interface
- URL/URI for addressing of Web pages (of modules of a Web app)
- Important to provide meaningful URLs
- Single URL (e.g. /app.cgi)
  - No bookmarks, difficult for humans, bad user interface
  - Problems with search engines

Hypertext engineering in Web applications (2/2)

- Multiple URLs
  - /student/add /student/show
  - /student/exam/add /student/exam/show
  - Meaningful, easier for humans
  - Bookmarks
  - Search engines can retrieve different parts and index it

Application logic (1/3)

- Traditionally, app logic manages the app state
  - E.g., the current state of the data, user inputs, etc.
- Typically, Web browser supports only HTML and does not have direct connection to the app logic
  - Communication over network and HTTP with the app logic
  - HTTP stateless (connection-less)
  - Web server needs to track users and sessions (cookies, URL rewriting)

Application logic (2/3)

- However, Web server provides only low-level tracking
  - Responsibility of the app logic to manage sessions
  - Manage it within the application server
  - App server has other responsibilities as well
  - Provides the basic app functionality
  - Communicates with data management modules
  - Communicates with external functionality (Web services)

Application logic (3/3)

- We need to take care of separation of concerns
  - Separate user interface and session management from the app logic
  - Separate user interface and session management from data management
  - ...
  - Design patterns

Data Management (1/6)

- Web server, application server, Web services
- All manage data → design patterns to separate concerns
- Disparate and numerous data sources
  - File system, databases, interlinked documents, document formats, metadata, etc..
  - Design patterns to abstract access to data sources
Typically, Web applications deal with relational databases. Need to manage relational data in object-oriented applications. Use design patterns like Data Access Object (DAO). Use object/relational mapping, like Hibernate framework.

Information retrieval
- How to find what I’m looking for?
- One approach are search engines with full-text processing
- Another approaches manage links
- Links in databases, or within documents
- Mixed approach: full-text and links, e.g. Google

Yet another approach is managing metadata
- Metadata is data about other data
- On the web
  - Tag information items (everything that you can access via URL) in a structured manner
- Search inside metadata

XML as a technology for managing data
- We need storage tools, query languages (XPath, XQuery)
- XML as a technology for exchanging data
  - E.g. export/import data from databases
- XML as a technology for distributed computing
  - E.g. exchange messages between Web services

Further Readings(1/3)
- General info on Web engineering
  - Web Engineering: An Introduction
    http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=19981&puNumber=93
  - Web Engineering: Creating a Discipline among Disciplines
    http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=19845&puNumber=93

Further Readings(2/3)
- Article on architecture of Web applications
  - Grady Booch: The Architecture of Web Applications
  - Grady Booch is one of the creators of UML

Further Readings(3/3)
Introduction to Frameworks for Exercises

Denis Helic

Three Java Frameworks + One Ruby Framework

Cocoon Framework

- Java Framework
- Completely based on XML and XML processing
  - I.e., you can develop your example completely in XML without procedural programming
- Based on principles of separating concerns
  - Complete separation of content, presentation, style, web-site management
  - Extensive use of XML stylesheet technologies, XSLT, XSL-FO

Cocoon Framework(1/6)

Cocoon Framework(2/6)

- Pipeline-based processing of XML
- XML Source → XML Transformation → XML Serializer
  - Source, e.g., an XML file
  - Transformation, e.g., an XSLT Transformation onto XHTML
  - Serializer sends back obtained XHTML

Cocoon Framework(3/6)

- Many different combinations are possible
- Alternative transformers
  - XSLT transformation onto SVG (interactive multimedia presentations)
  - XSLT transformation onto XSL-FO (formatting objects, “advanced CSS”)
- Alternative serializers
  - PDF serializer (in combination with XSL-FO)
  - Office serializer, etc.

Cocoon Framework(4/6)

- Alternative XML sources
  - Native XML databases
  - RDBMS, i.e., SQL sources
- Multiple Sources
  - Multiple XML Sources → XML Aggregators → XML Transformation → XML Serializer
  - Aggregators are typically XSLT stylesheets that filter and combine XML data

Cocoon Framework(5/6)

- Web-site management in a central file
  - Here you define all URLs of your Web application (matchers)
  - Using regular expressions to define sets of URLs, e.g. /student/*xml
  - Each matcher is related to an XML processing pipeline

Cocoon Framework(6/6)

- Download [http://cocoon.apache.org/mirror.cgi](http://cocoon.apache.org/mirror.cgi)
  - Only available as source distribution
  - Compile and deploy it as a Tomcat webapp
  - Example: [http://coronet.iicm.edu/cocoon/mmis2/](http://coronet.iicm.edu/cocoon/mmis2/)
### Struts Framework (1/5)

- Java Server-Side Web Framework
- Follows a MVC design pattern to separate content, logic, presentation
- Controller in Struts is already available
  - You need to configure it
- Model is a collection of application specific Java classes
  - You need to implement them
- View is a collection of JSP pages
  - You need to implement them

### Struts Framework (2/5)

- **Controller**
  - The server receives different HTTP requests
    - Requests can include different parameters submitted by the user
    - The server dispatches the request to a specific handler (action)
    - The server has a registry of mappings of parameters onto actions
    - The registry is a Struts configuration file: struts-config.xml
    - Action that handles the request is a subclass of a Struts Action class
    - You need to implement a number of action subclasses that handle requests

### Struts Framework (3/5)

- **Model**
  - Your application logic and application content is in a number of Java classes
  - Additionally, you may connect to a database to manage the app data
  - O/R mapping, data management patterns

### Struts Framework (4/5)

- **View**
  - When the action is executed the request is forwarded to a specific JSP page
  - JSP page reads the content from the Model (Java objects)
  - Produces valid HTML
  - HTML is presented to the user
  - Request → Find Action → Execute Action (Update Model) → Find View

### Struts Framework (5/5)

- **Download**
  - [http://struts.apache.org/download.cgi](http://struts.apache.org/download.cgi)
  - You can download either binaries or sources
  - In the installation included: struts-blank.war
    - You can use it for a quick start with Struts
    - Only edit struts-config.xml and define your own actions
    - Everything else is in place
  - Implement the model
  - Example: [http://coronet.iicm.edu/struts-publicationdb](http://coronet.iicm.edu/struts-publicationdb)

### Ruby on Rails Framework (1/4)

- Ruby is an object-oriented dynamic programming language
  - Similar to Smalltalk, Python
- Rails is an MVC Web Framework
- Principles behind Rails
  - Less software (a lot of source code is generated - data driven)
  - Convention over configuration (e.g., follow naming conventions and no need to configure anything)

### Ruby on Rails Framework (2/4)

- Typical RoR workflow
  - Create a database schema
    - Follow convention about naming tables and columns
  - Invoke a Rails script to generate model classes
  - Invoke a Rails script to generate controller classes
  - Start Rails server
  - Open your browser, access your app, and enjoy!

### Ruby on Rails Framework (3/4)

- Automatically generated insert, update, list controllers, views
- Model classes directly connected to a RDBMS
  - You can edit the generated source code and adjust it to your needs
    - e.g., create an association between two model classes (database relations)
- You can update the generated HTML templates
Ruby on Rails Framework


- Very productive
  - E.g., a small Web app with 2 related tables only 50 LOC
- Disadvantage: you need to learn Ruby
- I personally recommend it: you will for sure have fun with Rails!
  - Educational purpose: very clean implementation of MVC

Axis Framework

Java Web Services Framework

- Publishing of Web Services
  - You have a number of Java classes (e.g., your app logic)
  - You want to make a subset of that functionality available via Web
  - Inform Axis about which methods you want to offer
  - Automatically generate and publish it!

Clients communicate with services by exchanging messages in XML

Axis supports also automatic generation of client Java classes

Use this classes to work transparently with a Web service

- You write your client in Java and do not even know or care that the functionality is accessed over the Web
- You can write Rich Web clients, i.e., Java desktop apps, Java applets, etc.
- You can also try to write an AJAX client

Apache Axis:

- Example: [http://coronet2.iicm.edu/axis/services/persondao?method=readAllPersons](http://coronet2.iicm.edu/axis/services/persondao?method=readAllPersons)

XML Validation - Definition

- [XML well-formedness](#)
  - An XML document conforming to the basic XML syntax
  - Obviously, we need this to separate markup from content
- [XML validation](#)
  - An XML document conforming to a particular schema
  - The schema can be written in different schema languages
  - DTD, Relax NG, W3C XML Schema
- Why do we need the validation (we already have the well-formedness)?
  - To protect the system from data it cannot handle
  - E.g. a student matricel number is equal to zero
  - To enforce a contract between applications on the Web
  - For data exchange
  - Similar to an API, but here it is a contract for structuring data
  - Also similar to a database schema
- The process of checking the conformance is validation
  - There are different levels of validation
  - The validation of the markup (the content model validity)
    - The structure of an XML document
      - E.g. proper nesting of elements, proper attributes, etc.
The validation of values (datatypes)
- Does the content of an element have the proper datatype?
- Does an attribute have the proper datatype?
- Are the values of an element in a valid range?

The validation of integrity
- Links between nodes within a document (not that hard)
- Links between documents (very hard)

The validation of the semantics - business rules
- If the element “foo” contains string “bar” then the document must contain a “fred” element
- DTD, Relax NG, W3C XML Schema can not validate business rules
- You need a separate semantics-enabled validator
  - Your application logic in e.g. Java
  - Rule ML (not part of the course)
  - XSLT with XPath, e.g. Schematron (not part of the course)

Course XML document
- An XML document describing the MMIS2 course
- An extended version of the MMIS1 course example to depict validation

Content element can have a number of topic elements
- "XML Validation"
- "Data-oriented XML"
- "Web Services"
- "Semantic Web"

Schedule element contains a number of events
- Each event has a date, time, duration, place, etc.

Limitations of DTD
- Not in XML - need a separate parser
- Problems with namespaces, i.e., combining XML applications
- Poor support for datatypes, no integers, strings, etc.
- No datatypes at all for elements
- Especially important for data-oriented XML
- Syntactical limitations
  - A Calendar DTD with months and exact number of day-elements for each month!

Up to here the validation is purely syntactical
- What are the names of elements and attributes?
- What order they can appear, where they can appear?
- What are the spelling rules for attribute values and element content (data types)?
We have content models
We have datatype checks
- Day values are integers, time values are in a specific time format
- Duration values are integers in a range from 45 to 90
- Place values are of the form “HSi” + two digit number

We have business rules (very simple)
- Depending on the type of a date, sub-elements are structured differently

Complete document
http://coronet.iicm.edu/mmis2/examples/xml/validation/course.xml

DTD for the document (very limited functionality)
http://coronet.iicm.edu/mmis2/examples/xml/validation/course.dtd

A particular XML schema is a definition of a number of elements and attributes
- Each element or attribute has a name (tag-name) and a type
- Types can be complex types and simple types
- Complex types are used to define the content model
- Simple types are used to define datatypes of elements or attributes

 Basically, defining an XML schema is equal to defining types
- Types can be then referenced in element or attributes definitions
- You can think of a type as a class in object-oriented languages
- XML Schema allows to sub-class types
- In this way you can achieve more specific behaviour

http://www.w3.org/XML/Schema
Two parts of the recommendation
- XML Schema Part 1: Structures
  http://www.w3.org/TR/xmlschema-1/
- XML Schema Part 2: Datatypes
  http://www.w3.org/TR/xmlschema-2/

Defining a content model (Course element)

```xml
<xs:element name="course">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="courses:title"/>
      <xs:element ref="courses:description"/>
      <xs:element ref="courses:homepage"/>
      ... 
      <xs:element ref="courses:content"/>
      <xs:element ref="courses:goal"/>
      <xs:element ref="courses:schedule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
The course element is a complex type that has a sequence of elements. Elements appear in that order and are repeated once. minOccurs, maxOccurs attributes can be used. Default value is 1, unbound for as many as you want. Other elements are referred to (reuse of types).

The description element is of type string. The homepage is a complex type because it has a structure. The structure is composed of a textual content and two attributes. These elements are referred to from the course element.

The duration and the place element are user-defined types.

What is with the business rules? There is no means to check business rules in W3C XML Schema. Additional validator needed, e.g. XSLT and XPath. The Schematron:

http://xml.ascc.net/resource/schematron/schematron.html

Complete schema:
http://coronet.iicm.edu/mmis2/examples/xml/validation/course.xsd

Defining datatypes:
<xs:element name="description" type="xs:string"/>
<xs:element name="homepage">
  <xs:complexType mixed="true">
    <xs:attribute ref="xlink:href" use="required"/>
    <xs:attribute ref="xlink:type" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="lecturer" type="xs:string"/>

Defining datatype restrictions:
<xs:element name="time" type="xs:time"/>
<xs:element name="duration" type="courses:lectureDuration"/>
<xs:element name="place" type="courses:lectureRoomType"/>

The previous example shows also sub-typing. The new type (e.g. lectureRoomType) inherits properties from its supertype. It is also a string. Additionally it restricts the supertype.

The time element is of type xs:time:
http://www.w3.org/TR/xmlschema-2/

What is with the business rules?
<xs:element name="date">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="courses:day"/>
      <xs:element ref="courses:month"/>
      <xs:element ref="courses:year"/>
    </xs:choice>
    <xs:attribute name="type" use="required" type="xs:NCName"/>
  </xs:complexType>
</xs:element>
W3C XML Schema - Analysis

- Considered complex
- Not that flexible (structurely)
- Not that intuitive (types-based not structure-based)
- Long time needed to learn all the features
- Tool supported not that good as for DTD
  - But it is fastly growing

Relax NG

- Relax NG: Regular Language Description for XML Core - New Generation
  - Developed by OASIS (Organization for the Advancement of Structured Information Standards)
  - Developed as an ISO standard
  - Relax NG at Oasis

Relax NG - Features 1/3

- A schema written in Relax NG is close to a textual description of vocabulary
  - E.g. The course element contains the description element, the homepage element,...
  - E.g. The schedule element contains a number of event elements
    - An event element contains the date, time, place...
  - Wrap this in an XML-based syntax → A Relax NG schema

Relax NG - Features 2/3

- Complete separation between structure (the content model) and datatypes
  - Compare with W3C XML Schema where the structure is a (complex) datatype
  - The structure defined by means of patterns
    - Similar to regular expression patterns
    - But here patterns are patterns of elements, attributes
  - Reusing of patterns is possible
  - Very intuitive!

Relax NG - Features 3/3

- How datatypes are handled?
  - Specifies only few datatypes
    - You can plug-in any datatype system that you want
    - Mostly, W3C XML Schema datatype system

Relax NG - Example 1/6

- Defining a content model (Course element)
  - Just transform the above English sentences into XML
    <element name="course">
      <ref name="title"/>
      <element name="description">
        <text/>
      </element>
      ...
      <element name="lecturer">
        <text/>
      </element>
    </element>

Relax NG - Example 2/6

- Repeating elements
  - <element name="schedule">
    <oneOrMore>
      <element name="event">
        <element name="date">
          <attribute name="type">
            <choice>
              <value>Euro</value>
              <value>ISO</value>
              <value>US</value>
            </choice>
            ...
          </attribute>
        </element>
      </element>
    </oneOrMore>
  </element>

Relax NG - Example 3/6

- Reusing patterns (the title element)
  - Can be a part of the course and a part of an event
    <define name="title">
      <element name="title">
        <text/>
      </element>
    </define>
  - This is not a type it is a pattern to look for in an XML document
Handling datatypes

```xml
<grammar xmlns:xlink="http://www.w3.org/1999/xlink"
    ns="http://coronet.iicm.edu/courses"
    xmlns="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

    ... 

    <element name="year">
        <data type="integer"/>
    </element>

    ... 

</grammar>
```

Imposing restrictions on datatypes

```xml
<grammar xmlns:xlink="http://www.w3.org/1999/xlink"
    ns="http://coronet.iicm.edu/courses"
    xmlns="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

    ... 

    <element name="duration">
        <data type="integer">
            <param name="minInclusive">45</param>
            <param name="maxInclusive">90</param>
        </data>
    </element>

    <element name="place">
        <data type="string">
            <param name="pattern">HSi\d{2}</param>
        </data>
    </element>

</grammar>
```

Can not impose business rules (Interleave means in any order)

```xml
<interleave>
    <element name="day">
        <data type="integer"/>
    </element>

    <element name="month">
        <data type="string"/>
    </element>

    <element name="year">
        <data type="integer"/>
    </element>

</interleave>
```

Complete example

http://coronet.iicm.edu/mmis2/examples/xml/validation/course.rng

### Schemas development tools (1/2)

XMLSpy (Altova from Vienna)
http://www.altova.com
Validates with DTDs, W3C XML Schemas

Jing by James Clark
http://www.thaiopensource.com/relaxng/jing.html
Validates with Relax NG

Schema converter Trang by James Clark
http://www.thaiopensource.com/relaxng/trang.html
Converts in all possible directions between DTD, Relax NG, XML Schema

### Schemas development tools (2/2)

Using Trang to do Validation by Instance
First you write an XML document
You work with the document and improve it
Then you use Trang to automatically obtain a schema (e.g. Relax NG)
Then you optimize the schema manually
Very productive way to develop schemas

### Further Readings

Schema articles on www.xml.com
http://www.xml.com/pub/q/all_schema

W3C XML Schema Tutorial
http://www.xfront.com/

Relax NG Tutorial
http://www.oasis-open.org/committees/relax-ng/tutorial.html

### XML Publishing

Denis Helic
Different data sources in Web applications
- Documents in file systems (HTML, XML, SVG, ...)
- Data in relational and XML databases
- Metadata in different formats (files, databases,...)

Web-based XML publishing approach
- Applies XML as a common denominator (gemeinsamer Nenner)
- Bring all data in at least XML
  - Dynamically

Three layers of data management
- The first (low-level) layer deals with data in a storage-specific way
  - Data Access Layer
    - Access files in a file system by means of File I/O
    - Use SQL to access RDBMS, e.g., programmatically with JDBC in Java
- The second layer deals only with XML
  - Data Aggregation Layer
    - Output of the first layer is input for the second layer
    - Integrating (aggregating) data from different sources
    - Once when you have all data sources in XML → easy integration
    - Output of the second layer is input for the third layer
- The third layer deals with presentation of XML
  - Data Presentation Layer
    - Presenting in different formats such as PDF, XHTML, SVG
    - Presenting for different devices such as desktop PC, PDA, mobile phone, etc.
    - Applying XSL-T and XSL-FO to do the job
We have a Web application offering information about courses and students.
Which courses do I have, which students registered for what course?
An 1:n relation between courses and students.
Information can be presented in HTML for viewing, browsing.
Additionally, presentation as PDF for e.g. printing.

Manage course information in an XML document.
MMIS 2 course XML document http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/content/course.xml
Advantages of managing course information as XML.
Single-source publishing is available.
Manage document only in XML, publish in different formats.

You can impose XML validation during content authoring.
Using an XML editor that supports validation.
Authors (lecturers) must create the documents that you expect!
The first layer only retrieves the document from the file system.

Manage students data in a relational database.
Data is updated often (e.g. new students, new term, ...)
Query the database by means of SQL, e.g. in Java.
Wrap the data into an XML document (the first layer).

The second layer has all the data available in XML.
Validation, integration of the data in a single chunk of XML data.
The third layer can present the data in XHTML or in PDF.

The second layer has all the data available in XML.
Validation, integration of the data in a single chunk of XML data.
The third layer can present the data in XHTML or in PDF.

Coocon - Java-based XML Publishing Framework (1/19)
An Open Source Apache project http://cocoon.apache.org/
Web development framework supporting separation of concerns http://cocoon.apache.org/2.1/introduction.html
Management of a Web-site.
Based on the concept of component pipelines

A chunk of XML data is pushed through a pipeline

Matcher, generator, transformer, serializer

Corresponds approximately the layers from above

Transformer sometimes receives something else than XML

Very often transformer produces the presentation

Since everything in XML is not so dangerous

Matchers

General

Pattern

Wildcards, regular expressions

When the URL is matched it dispatches the request to a particular pipeline

The pipeline processes the XML document and produces a response

Generators

Default generator is the FileGenerator

SAX is Simple API for XML

Originally Java, event-based

Generators

Creates XML data from a data source

A lot of generators in Coocoon

Default generator is the FileGenerator
Transformers
Transforms XML data coming from a generator into another XML
Using XSLT, XSLTTransformer is default transformer
A lot of transformers in Coocoon
http://cocoon.apache.org/2.1/userdocs/transformers.html

Note SQLTransformer
Recieves response from the RDBMS
Transformes that response into Cocoon internal XML
Chain another transformer to get it into XML format that you want

You can chain transformers to make more than one transformation
<map:match pattern="students.html">
  <map:generate src="content/students.xml"/>
  <map:transform type="sql">
    <map:parameter name="use-connection" value="mmis2"/>
  </map:transform>
  <map:transform src="style/xsl/sql2xml.xsl"/>
  <map:transform src="style/xsl/students.xsl"/>
  <map:serialize type="html"/>
</map:match>

What is the generator for the SQLTransformer?
It is an XML file containing an SQL query
<page xmlns:sql="http://apache.org/cocoon/SQL/2.0">
  <content>
    <execute-query xmlns="http://apache.org/cocoon/SQL/2.0">
      <query>
        select * from registered
      </query>
    </execute-query>
  </content>
</page>

Web-site management
All pipelines defined in a central file called sitemap.xmap
Another XML file :) 
One sitemap.xmap per directory (similar to .htaccess)
You can access only the URLs defined in the sitemap.xmap
Top sitemap.xmap contains general configuration
Inherited by all sitemaps in sub-directories

Serializer
Writes the XML data in a specified format
Default is HTML
Many serializers in Cocoon
http://cocoon.apache.org/2.1/userdocs/serializers.html

Note SQLTransformer
Recieves response from the RDBMS
Transformes that response into Cocoon internal XML
Chain another transformer to get it into XML format that you want

<map:match pattern=""/>
<map:transform src="context://mmis2/style/xsl/html/simple-samples2html.xsl">
  <map:parameter name="contextPath" value="{request:contextPath}"/>
</map:transform>

... }
</map:match>

<map:match pattern="">
  <map:generate src="examples.xml"/>
  <map:transform src="context://mmis2/style/xsl/simple-samples2html.xsl">
    <map:parameter name="contextPath" value="{request:contextPath}"/>
  </map:transform>
</map:match>

<map:pipelines>
  <map:pipeline>
    <map:match pattern="course.xml">
      ...
    </map:match>
    ...
    <map:match pattern="course.pdf">
      <map:generate src="content/course.pdf"/>
      <map:transform src="style/xsl/course2fo.xsl"/>
      <map:serialize type="fo2pdf"/>
    </map:match>
    ...
  </map:pipeline>
</map:pipelines>
</map:sitemap>

<map:match pattern="">
  <map:generate src="content/students.xml"/>
  <map:transform src="style/xsl/students.xsl"/>
  <map:serialize type="html"/>
</map:match>
Where is the logic?

- XSP (eXtensible Server Pages)
  - It is an XSLT document that contains Java code
- Actions (under development)
  - Java code
    - [http://cocoon.apache.org/2.1/userdocs/actions.html](http://cocoon.apache.org/2.1/userdocs/actions.html)
  - Within XSLT

Cocoon - Example(1/7)

- MMIS 2 course and registered students in Cocoon
  - [http://coronet.iicm.edu/cocoon/mmis2/](http://coronet.iicm.edu/cocoon/mmis2/)
- Samples XML document from Cocoon
  - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/examples.xml](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/examples.xml)
- XSLT for transforming it into HTML already available
  - [sitemap.xmap](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/sitemap.xmap)

Cocoon - Example(2/7)

- Example has three parts
  - The first part: Course XML document
    - XSLT for producing HTML
      - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/xsl/course.xsl](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/xsl/course.xsl)
    - CSS for formatting HTML
      - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/css/course.css](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/css/course.css)
  - How to retrieve data from MySQL: sitemap.xmap
    - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/content/students.xml](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/content/students.xml)

Cocoon - Example(3/7)

- The second part: the student database
  - Registered students database from MMIS 1
    - Inserting students
      - [http://coronet.iicm.edu/mmis/examples/java/mysql/form.html](http://coronet.iicm.edu/mmis/examples/java/mysql/form.html)
    - Retrieving students
      - [http://coronet.iicm.edu/mmis-servlets/Registration](http://coronet.iicm.edu/mmis-servlets/Registration)

Cocoon - Example(4/7)

- XSLT for producing HTML
  - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/style/xsl/students.xsl](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/style/xsl/students.xsl)
- How to retrieve data from MySQL: sitemap.xmap
  - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/sitemap.xmap](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/sitemap.xmap)
- SQL document
  - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/content/students.xml](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/content/students.xml)

Cocoon - Example(5/7)

- Additionally, load the MySQL driver in web.xml
  - `<init-param>`
    - `<param-name>`load-class`</param-name>`
    - `<param-value>`org.gjt.mm.mysql.Driver`</param-value>`
  - `<param-name>`load-class`</param-name>`
  - `<param-value>`org.gjt.mm.mysql.Driver`</param-value>`

Cocoon - Example(6/7)

- Define SQL connection (Cocoon manages connection pool) in co-con.xconf
  - `<datasources>`
    - `...`
    - `<jdbc name="mmis2">`
      - `<pool-controller max="10" min="5"/>
      - `<auto-commit>false</auto-commit>`
      - `<dburl>jdbc:mysql://localhost/mmis</dburl>`
      - `<user>student</user>`
      - `<password>student</password>`
    - `</jdbc>`
    - `...`
  - `</datasources>`

Cocoon - Example(7/7)

- The third part: data integration
  - How to integrate data: sitemap.xmap
    - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/sitemap.xmap](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/sitemap.xmap)
- XSLT for producing HTML
  - [http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/style/xsl/course.xsl](http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/style/xsl/course.xsl)
- Reusing of stylesheets from before
XSL Formatting Objects: XSL-FO
The second part of XSL specification
http://www.w3.org/TR/xsl/
Similar to CSS
You can define font-family, font-weight
An XML syntax and more powerful than CSS

Usually you will not create XSL-FO sheet manually
You use XSLT to produce XSL-FO from your input document
You need a processor (renderer) to translate an XSL-FO into PDF
Apache FOP
http://xml.apache.org/fop/
Integrated into Cocoon

Page properties
<fo:layout-master-set>
<fo:simple-page-master margin-right="2.5cm" margin-left="2.5cm" margin-bottom="2cm" margin-top="1cm" page-width="21cm" page-height="29.7cm" master-name="page">
<fo:region-before extent="3cm"/>
<fo:region-body margin-top="3cm"/>
<fo:region-after extent="1.5cm"/>
</fo:simple-page-master>
</fo:layout-master-set>

Page numbering
<fo:page-sequence master-reference="all">
<fo:static-content flow-name="xsl-region-after">
<fo:block line-height="14pt" font-family="serif" font-size="10pt" text-align="center">
page
<fo:page-number/>
</fo:block>
</fo:static-content>

Page flow within blocks
<fo:flow flow-name="xsl-region-body">
<fo:block line-height="14pt" font-family="serif" font-size="36pt"> Multimedia Information Systems 2 </fo:block>
...
<fo:block line-height="14pt" font-size="14pt"> General course information </fo:block>
...</fo:flow>

Links
<fo:block text-align="justify" space-before.optimum="12pt" space-after.optimum="12pt" font-size="12pt"> Homepage of the course: MMIS2 homepage </fo:block>

Lists
<fo:list-block font-size="12pt"> MMIS 2 course and registered students in Cocoon http://coronet.iicm.edu/cocoon/mmis2/ Produce PDF out of Course XML document Some features of XSL-FO explained </fo:list-block>
XSL-FO Cocoon example(7/8)

```xml
<fo:table>
  <fo:table-header>
    <fo:table-row>
      <fo:table-cell>
        <fo:block>Date</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-header>
  ...
</fo:table>
```

Further Readings

- Style articles on [www.xml.com](http://www.xml.com/style/)
- Java and XML articles on [www.onjava.com](http://www.onjava.com/topics/java/java_xml)

XML Data Management - Introduction(1/4)

- How do we store XML data?
  - As XML documents in the file system
  - As data in RDBMS that is mapped onto XML documents
    - Structured data → easy to separate content and structure (database schema)
    - Each data item (e.g., row in a table) has the same structure

XML Data Management - Introduction(2/4)

The MMIS 2 course example
- Course XML document: [http://coronet.iicm.edu/coocoon/mmis2/course/course.xml](http://coronet.iicm.edu/coocoon/mmis2/course/course.xml)
- Structured data about registered students (mapped onto XML) [http://coronet.iicm.edu/coocoon/mmis2/course/students/students.xml](http://coronet.iicm.edu/coocoon/mmis2/course/students/students.xml)
- Very simplified example, since a single document and a single table in the database

XML Data Management - Introduction(3/4)

- How real-life Web applications look like?
  - Hundreds, thousands, even millions of documents
  - Frequent management operations, e.g., add new, delete, update, query, ...
  - Multiple databases with hundreds or thousands of tables
  - Again frequent management operations

XML Data Management - Introduction(4/4)

- What are the consequences of this situation?
  - Scalability, performance, reliability, data access, ...
  - DBMS excel regarding these issues
  - File system has tremendous problems
Data-centric XML(1/2)

- Representing structured data in XML
- Each data item has the same structure
- Important to be able to handle various data types
  - Integers, strings, floats, dates, ...
- Order of XML elements within a data item has no semantic significance
- Registered students in MMIS 2
  - http://coronet.iicm.edu/cocoon/mmis2/course/students/students.xml

Data-centric XML(2/2)

- DBMS are very good at management of structured data
- Mapping onto XML is straightforward
- Ordering is not important
- Map a table (or query results) onto single XML document
- Always use DBMS (especially RDBMS) for management of structured data

Document-centric XML(1/2)

- Representing semi-structured (e.g. documents) data in XML
- Data items can have different structure
  - A list in XHTML has a different structure then a paragraph
  - Even two paragraphs can have different structure (e.g. different sub-elements)

Document-centric XML(2/2)

- It is not important to handle different data types
  - Typically textual content
- Ordering of elements has semantic significance
- XHTML is just a single document format, SVG, MathML, ...

Managing Document-centric XML: Alternative 1(1/6)

- Store XML documents in the file system
- MMIS 2 course example
  - Course XML document contains calendar of events
    - http://coronet.iicm.edu/cocoon/mmis2db/course/course.xml
  - Resources XML document contains description and links to external resources
    - http://coronet.iicm.edu/cocoon/mmis2db/course/resources.xml

Managing Document-centric XML: Alternative 1(2/6)

- A dynamically created description of the current event
- The current event info
  - Retrieve it from the course XML document
- The description and links of resources for the current event
  - Retrieve it from the resources XML document

Managing Document-centric XML: Alternative 1(3/6)

- We need to process both XML documents
- Processing the documents includes
  - Load the documents
  - Parse the documents
  - Retrieve the needed info with XPath
  - Present it with XSLT
- Scalability problems, performance problems, slow data access, ...

Managing Document-centric XML: Alternative 1(4/6)

- Example with Cocoon
  - http://coronet.iicm.edu/cocoon/mmis2db
- XSLT for producing Course and combined HTML
  - http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/course.xsl
- XSLT for producing Resources HTML
  - http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/resources.xsl
- Sitemap Matchers
  - http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/sitemap.xmap
Managing Document-centric XML: Alternative 1(5/6)

XSLT for querying documents
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/event.xsl

...<xsl:param name="current_event"/>
...

<xsl:template match="courses:event">
  <xsl:if test="normalize-space(courses:title)=$current_event">
    <h2>
      Lecture on <xsl:value-of select="courses:title"/>
    </h2>
  </xsl:if>
</xsl:template>

Managing Document-centric XML: Alternative 1(6/6)

How to pass request parameters to an XSLT style
...
<map:transform src="style/xsl/event.xsl">
  <map:parameter name="use-request-parameters" value="true"/>
</map:transform>
...

Sitemap Matchers
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/sitemap.xmap

Managing Document-centric XML: Alternative 2

Store XML data in a DBMS (e.g., RDBMS)

- How to map semi-structured data onto structured data storage?
- How to keep ordering of elements?
- Multiple tables, multiple joins to ensure correct mapping
- Bad performance, not reliable because of errors!

Managing Document-centric XML: Alternative 3(1/3)

Queries
- Give me the third sub-element of the element with the name 'schedule'
- Give me the attribute value of the attribute with the name 'href' of the first element
- Give me the second processing instruction from this document

Managing Document-centric XML: Alternative 3(2/3)

Updates
- Store these elements (in exactly this order) as sub-elements of the 'course' element
- Insert this processing instruction into this document
- Add this document
- Delete this document

Managing Document-centric XML: Alternative 3(3/3)

Supporting XML data model with database technology

- To improve scalability, performance, data access, reliability
  - Use database technology to support XML data model natively
  - Native XML interface
  - Two ways to provide such a support
    - XML-enabled databases (At some stage similar to Alternative 2)
    - Native XML databases

Native XML databases - Definition(1/2)

- Defines a (logical) model for an XML document
  - Stores and retrieves documents according to that model
  - Must include elements, attributes, PCDATA, and document order
  - e.g., XPath, DOM, SAX, etc.
Native XML databases - Definition

- Has an XML document as its fundamental unit of (logical) storage
  - e.g. a table (relation) in a RDBMS
- It is not required to have any particular underlying physical storage model
  - Built on relational, OO databases, indexed files!
- Definition comes from XML:DB initiative

Native XML Databases - Collections

- Collections (sets) of documents
- Query all documents in the collection
- Namespace aware
- Indexed documents
  - Better performance, data access, etc.

Native XML Databases - Query Languages

- XPath
  - Advantages
    - If you know XSLT you know XPath
    - Intuitive: file system wildcards, patterns, etc.
  - Disadvantages
    - Grouping, sorting, cross document joins,
    - Need to be a part of a more powerful language, e.g. XSLT
- Recently XQuery
  - XQuery at W3C
  - http://www.w3.org/XML/Query
  - Working draft status, recommendation expected
  - Not only a query language but also a fully-fledged programming language
  - Host language for XPath

Native XML Databases - Query Languages

- So-called FLWOR (“flower”) expression
  - For-Let-Where-Order-Return
  - With return statement you produce resulting XML
  - You can produce directly HTML
  - Hence no need for XSLT

for $d in document("depts.xml")//deptno
let $e := document("emps.xml")//employee[deptno = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
<big-dept>
  { $d,
    <headcount>{count($e)}</headcount>,
    <avgsal>{avg($e/salary)}</avgsal>
  }
</big-dept>
Native XML Databases - Query Languages

- Comparison with XSLT
  - Matter of personal preference
  - Procedural vs. functional programming

Native XML Databases - Query Languages

Some problems more easily solved with recursion (e.g. a lot of mixed content)

```xml
<xsl:template match="resources:resource">
  <li>
    <xsl:apply-templates/>
  </li>
</xsl:template>
<xsl:template match="resources:link">
</xsl:template>
```

Native XML Databases - Query Languages

Some problems more elegant with procedural programming (e.g. if-else-then statement)

```xml
<xsl:choose>
  <xsl:when test="courses:slides/@xlink:href">
  </xsl:when>
  <xsl:otherwise>
    <xsl:value-of select="courses:slides"/>
  </xsl:otherwise>
</xsl:choose>
```

Native XML Databases - Updates

- Weakness of native XML databases
- Specific solutions by different vendors
- Retrieve whole document, modify, insert again
- XUpdate initiative
- W3C works on update functionality for XQuery
  - Will take some time

Native XML databases - Example

- Example created with two native XML databases and Cocoon
- Xindice from Apache
  - http://xml.apache.org/xindice/
  - Query language: XPath
- eXist
  - http://exist.sourceforge.net/
  - Query language: XQuery

Native XML databases - Example

- Xindice comes preinstalled with Cocoon
  - Installed as a Cocoon block (plug-in)
- Data management with a command line tool
- Uses XML-RPC to connect to the database servlet
  - We will talk about it in Web services lecture

Native XML databases - Example

- Example for MMIS 2 course installed in Cocoon
  - http://coronet.iicm.edu/cocoon/mmis2db/course
  - Specify a title of an event to retrieve the info on that event
  - The info comes from two documents stored in Xindice database
    - Course XML document: the info about date, place, ...
    - Resources XML document: the info about links
Native XML databases - Example(4/14)

Query Course XML document
<map:match pattern="course_query.xml">
  <map:match type="request-parameter" pattern="xpath">
    <map:generate src="xmldb:xindice-embed:///db/mmis2_course/#{1}"/>
    <map:serialize type="xml"/>
  </map:match>
</map:match>

Native XML databases - Example(5/14)

Query both documents and integrate the results
<map:match pattern="event_query.xml">
  <map:match type="request-parameter" pattern="current_event">
    <map:aggregate element="aggregated-content">
      <map:part src="xmldb:xindice-embed:///db/mmis2_course/##courses:event[normalize-space(courses:title)='{1}']"/>
      <map:part src="xmldb:xindice-embed:///db/mmis2_resources/##resources:event[normalize-space(resources:name)='{1}']"/>
    </map:aggregate>
    <map:transform src="style/xsl/event_query.xsl"/>
    <map:serialize type="html"/>
  </map:match>
</map:match>

Native XML databases - Example(6/14)

XSLT for producing HTML
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/event_query.xsl

Native XML databases - Example(7/14)

Example for MMIS 2 course installed in Cocoon
http://coronet.iicm.edu/cocoon/mmis2db/course
Both documents in one eXist collection
Query both of the documents with an XQuery
Retrieve the info from both documents, integrate it and present it

Native XML databases - Example(8/14)
eXist native XML database
Does not come preinstalled with Cocoon
Install it as a Cocoon block
http://wiki.apache.org/cocoon/EXistAsBlock
Examples, administration available directly from Cocoon
http://coronet.iicm.edu/cocoon/samples/blocks/exist/index.xml
XQueryGenerator a special type of Cocoon generator

Native XML databases - Example(9/14)

XQueryGenerator in sitemap.xmap
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/sitemap.xmap

Native XML databases - Example(10/14)

Simple XPath query submitted
xquery version "1.0";
declare namespace util="http://exist-db.org/xquery/util";
declare namespace request="http://exist-db.org/xquery/request";
<results>
  { let $query := request:request-parameter("xpath", ())
    let $result := util:eval($query, "/db/mmis2")
    return $result
  }
</results>
Complete XQuery file
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/content/exist_query.xq

Native XML databases - Example(11/14)

XQueryGenerator in sitemap.xmap
<map:match pattern="exist_query.xq">
  <map:generate src="content/exist_query.xq" type="xquery"/>
  <map:serialize type="xml"/>
</map:match>

Sitemap.xmap
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/sitemap.xmap
Native XML databases - Example(12/14)

- Querying both documents for an event

```xml
<result>
{
...
... "//resources:event[resources:name = "," , $current_event,"']
let $result := util:eval($query, "/db/mmis2")
return $result
</result>
```

Complete XQuery file
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/content/exist_event_query.xq

Native XML databases - Example(13/14)

- Presenting results as XML or as HTML

```
<map:match pattern="exist_event_query.xq">
<map:generate src="content/exist_event_query.xq" type="xquery"/>
<map:serialize type="xml"/>
</map:match>
<map:match pattern="event_query.xq">
<map:generate src="content/exist_event_query.xq" type="xquery"/>
<map:transform src="style/xsl/event_query.xsl"/>
<map:serialize type="html"/>
</map:match>
```

XSLT for producing HTML
http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/event_query.xsl

Native XML databases - Example(14/14)

- Producing HTML directly from XQuery

```
<map:match pattern="event.xq">
<map:generate src="content/current_event.xq" type="xquery"/>
<map:serialize type="html"/>
</map:match>
```

Further Readings(1/2)

- XML Data Stores: Emerging Practices
- Introduction to Native XML Databases
  http://www.xml.com/pub/a/2001/10/31/nativexmldb.html
- Getting Started with XQuery (Part 1)
  http://www.xml.com/pub/a/2005/03/02/xquery.html

Further Readings(2/2)

- Getting Started with XQuery (Part 2)
- Database articles on xml.com
  http://www.xml.com/databases/
- Simple XQuery Tutorial

Web Application Frameworks - MVC

Denis Helic

Separation of Concerns(1/5)

- Core design pattern of all software engineering
  - Easily supported by object-oriented software development
  - Definition by Edsger Dijkstra

We know that a program must be correct and we can study it from that viewpoint only; we also know that is should be efficient and we can study its efficiency on another day [...] But nothing is gained - on the contrary - by tackling these various aspects simultaneously. It is what I sometimes have called “the separation of concerns” [...]“On the role of scientific thought”

Separation of Concerns(2/5)

- Edsger Dijkstra one of the most influential computer scientists
- Paper: Go To Statement Considered Harmful
- Algorithm: Shortest-path algorithm
- Book: A discipline of programming
- Award: The ACM Turing Award
You want to isolate different aspects of a software application from each other
- You can work on each aspect in details
- You can be consistent within each aspect
- At a single moment you work on a single aspect (e.g., you are not distracted with other aspects)
- Also, a basis for the team work (e.g., different teams work on different aspects)

Divide-and-conquer method of designing algorithms
- Applied to developing software applications
- OO principles easily support SOC
- Classes, objects, encapsulation isolate one aspect from another
- If interfaces are clearly defined you can work on different aspects in isolation

Benefits of SOC
- Improved comprehension of the application domain
- Reduced complexity
- Component (aspect) integration
- Reuse
- Adaptibility, customization (through component exchange)

The interface between UI and Data Management is crucial!
- It is where the SOC is violated most
- Why is this so?
- An example: You have a database containing info about the students registered for a course
  - This info is composed of name, matrikel number and study field

In your process logic you have a Student class
- The Student class has getter and setter methods
  - getName(), getStudyField(), etc.
- In your UI script (e.g., a PHP script) you retrieve a list of Student objects
  - You iterate through the list and use the getter methods to write info in an HTML table

You mix UI and data management
- Consequences can be tremendous: different UI scripts need to be updated, maintained, etc.
- Proper way of implementing such changes
  - Modify the Student class and extend it with getEMail(), setEMail()
  - In the DM module connect to the external database and populate student objects with setEMail()
SOC is supported by OO programming languages but it is not enforced!
Developers need to take care about this!
Scripting languages are even more vulnerable
You do not need compiling, it is fast to make such changes!
But, Java is vulnerable too, here you need to take care also!
Recollect, the clear solution that we had with XML publishing
The UI gets an XML document that can only be presented with XSLT!

One way to improve the situation: a layered architecture
The UI communicates only with the PL module
Again, hard to enforce this
You can only hope that developers will follow the principle!

Model-View-Controller is a particular design pattern that supports SOC
It was invented in the early days of GUIs
To decouple the graphical interface from the application data and logic
Invented at Xerox Parc in the 70’s

Controller
Handles user input (e.g., mouse clicks, keyboard,...)
Updates the model
Instructs the view to redraw itself
**View**
- Presents the model in a specific way
- Note different views for the same model
- Very important not only in GUIs but also in Web applications (e.g., XHTML, PDF, etc.)

**Model**
- Contains the data and application logic

**Very easily accomplished with an OO programming model**
- Objects encapsulate the data
- Objects implement behaviour (as methods)
- Interaction between different objects (i.e., invoking methods) supports the application logic

**Where does the data come from?**
- If in memory → everything is covered
- If in the file system or in a DBMS
  - We need a special Data Management module

**A special case of the MVC uses Observer design pattern**
- When the model changes it notifies the views about the change
  - All views redraw as the result of the notification
  - Not applicable in a Web application
    - Page-oriented applications
    - You need a user request for each particular view (e.g., HTML, PDF, ...)

**MVC: An example with Observer pattern**
- A simple GUI Java application using MVC
  - The example uses Observable/Observer pair from java.util
  - Subclasses of the Observable class can notify observers about changes in their state
  - Implementation classes of Observer need to implement update() method to react to the changes
  - Observer is a Java interface
MVC: An example with Observer pattern(2/11)

**Model**
- SimpleModel holding a single integer value
- `getValue()` and `setValue()` methods to manipulate the model
- The value must be in the range `[0, 100]`

```java
final public class SimpleModel extends Observable {
    private int value_;
    public void setValue(int value) {
        if ((value > 100) || (value < 0)) {
            throw new IllegalArgumentException("The value must be ..");
        }
        value_ = value;
        setChanged();
        notifyObservers();
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleModel.java

MVC: An example with Observer pattern(3/11)

**View**
- Holds a reference to the model
- Manages a GUI widget to represent the model value
- Reacts to change notifications from the model
- It is an abstract class, where subclasses manage different GUI widgets

```java
public abstract class SimpleView implements Observer {
    protected JComponent widget_; 
    public void update(Observable observable, Object arg) {
        updateView();
    }
    abstract public void updateView();
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleView.java

MVC: An example with Observer pattern(4/11)

```java
final public class SimpleTextFieldView extends SimpleView {
    private JTextField value_field_ = new JTextField();
    public SimpleTextFieldView(SimpleModel model) {
        widget_ = new JPanel(new BorderLayout());
        widget_.add(value_field_, BorderLayout.SOUTH);
        value_field_.addActionListener(new TextFieldControllerAction());
    }
    class TextFieldControllerAction implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            model_.setValue(Integer.parseInt(value_field_.getText()));
        }
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleTextFieldView.java

MVC: An example with Observer pattern(5/11)

```java
public abstract class SimpleSliderView extends SimpleView {
    private JSlider value_slider_ = new JSlider();
    public SimpleSliderView(SimpleModel model) {
        widget_ = new JPanel(new BorderLayout());
        widget_.add(value_slider_, BorderLayout.SOUTH);
    }
    public void updateView() {
        value_slider_.setValue(model_.getValue());
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleSliderView.java

MVC: An example with Observer pattern(6/11)

Controller
- Java Event System follows already MVC, i.e., it is the controller
- The event capturer forwards a captured event (e.g. a mouse click) to the event dispatcher
- The event dispatcher manages a mapping of events to event listeners
- The event dispatcher finds the proper listener and invokes its `actionPerformed()` method
- We need only to write listeners, register them with GUI widgets and implement `action()` methods

```java
final public class SimpleTextFieldView extends SimpleView {
    public SimpleTextFieldView(SimpleModel model) {
        ... value_field_.addActionListener(new TextFieldControllerAction());
        ...
    }
    class TextFieldControllerAction implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            model_.setValue(Integer.parseInt(value_field_.getText()));
        }
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleTextFieldView.java

MVC: An example with Observer pattern(7/11)

```java
final public class SimpleSliderView extends SimpleView {
    public SimpleSliderView(SimpleModel model) {
        ... value_slider_.setValue(model_.getValue());
        ...
    }
    public void updateView() {
        model_.setValueChanged();
        notifyObservers();
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleSliderView.java

MVC: An example with Observer pattern(8/11)

```java
final public class SimpleTextFieldView extends SimpleView {
    public SimpleTextFieldView(SimpleModel model) {
        ... value_field_.addActionListener(new TextFieldControllerAction());
        ...
    }
    class TextFieldControllerAction implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            model_.setValue(Integer.parseInt(value_field_.getText()));
        }
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleTextFieldView.java
MVC: An example with Observer pattern

```java
public class SimpleSliderView extends SimpleView {
    public SimpleSliderView(SimpleModel model) {
        ...
        value_slider_.addChangeListener(new SliderControllerAction);
        ...
    }
}
class SliderControllerAction implements ChangeListener {
    public void stateChanged(ChangeEvent event) {
        model_.setValue(value_slider_.getValue());
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleSliderView.java

MVC on the server side

The server receives different HTTP requests
- Requests can include different parameters submitted by the user
- On the basis of these parameters the server produces the response
- The server can dispatch the request to different handlers (actions)
- The server needs a registry of mappings of parameters onto actions
- The server, registry, dispatcher and actions are the Controller

```
public static void main(String[] args) {
    SimpleModel model = new SimpleModel();
    SimpleView view = new SimpleTextFieldView(model);
    ...
    view = new SimpleSliderView(model);
    ...
    model.setValue(12);
```

MVC on the server side

```
The server receives different HTTP requests
Requests can include different parameters submitted by the user
- On the basis of these parameters the server produces the response
- The server can dispatch the request to different handlers (actions)
- The server needs a registry of mappings of parameters onto actions
- The server, registry, dispatcher and actions are the Controller
```

Struts - Java-based MVC Web App Framework

An Open Source Apache project
http://struts.apache.org/
Web development framework supporting MVC and SOC
http://struts.apache.org/userGuide/introduction.html#mvc
It follows the design described above
- The controller is the Struts ActionServlet + Actions
- The model is your application data and logic
- The views are JSP pages
- Everything defined in struts-config.xml

```
Download
http://struts.apache.org/download.cgi
You can download either binaries or sources
In the installation included: struts-blank.war
- You can use it for a quick start with Struts
- Only edit struts-config.xml and define your own actions
- Everything else is in place
```

Struts - Java-based MVC Web App Framework

Defining actions in struts-config.xml has a number of steps
- Defining a URL pattern for that action (pattern + .do)
- Defining a so-called ActionForm which encapsulates the user parameters
- There is a special JavaBean class that extends ActionForm class
- This class provides getters and setters for the parameters (same names)
- Defining a number of forwards to JSP pages to present the results
If validate set to true then you can validate that the user parameters have the proper values.

Need to implement the validate() method inside your ActionForm subclass.

If validation fails

The user is redirected to JSP specified in the input attribute.

Name parameter refers to the ActionForm subclass

<form-beans>
  ...
  <form-bean
    name="search_form"
    type="edu.iicm.publication.struts.SearchForm"/>
  ...
</form-beans>

Additionally, Java classes need to be implemented

ActionForm subclass encapsulating parameters

A subclass of the Action class which manipulates the Model objects

Here you also decide to which JSP page to forward

Finally, a number of JSP pages to present the results

A special user as administrator

Can add authors

Can add publications

Delete, update!

Importing publications in different formats

BibTeX, XML

BibTeX format mapped onto a relational database (e.g., a number of relations)
Here we just want to investigate the Struts part of the example

Search form, submitting queries
- Results in different formats
- Administration interface
- Registering, validating inputs, etc.

Right now we have the Model

The Model includes classes for publications
- The Model and the DM are separated
  - The lecture on O/R mapping will show how this is done
- The Model
  - http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
    src/edu/iicm/publication/

Search functionality

```
<action
  path="/search"
  type="edu.iicm.publication.struts.SearchAction"
  name="search_form"
  scope="request"
  validate="false"
  input="/search.jsp">
  <forward name="html" path="/search_results.jsp"/>
  <forward name="bibtex" path="/search_export.jsp"/>
  <forward name="rdf" path="/search_export.jsp"/>
</action>
```

Action class

```
public final class SearchAction extends Action {
  public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    String type = ((SearchForm) form).getType();
    ...
    String format = ((SearchForm) form).getFormat();
    }
}
```

JSP views

```
<%@ include file="/jsp_layout/header.jsp" %>
<h2 class="hrthinbox">
All Publications
</h2>
<ul>
  <%= request.getAttribute("result").toString() %>
</ul>
<div>
  <a href = "search.jsp">Search again</a>
</div>
<%@ include file="/jsp_layout/footer.jsp" %>
```

The example

http://coronet.iicm.edu/struts-publicationdb

Search functionality
- You can select different criteria
- You can select different output formats
To have different output formats we use Visitor pattern
Publication is an abstract class with an abstract write method
Subclasses (Article, Book, ...) invoke an abstract method from an abstract Visitor
They pass the reference to itself to the visitor
Subclasses of Visitor implement the method
Write out the proper format

New format \rightarrow new Visitor
Creation of Visitor with Abstract Factory
Allows you to use the same code to write out HTML, BibTeX, ...
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/writer

Administration functionality
Adding, deleting, updating, importing publications
User management
The user needs to log-in
The user info need to be always available
The user info should be used to check the rights

Session management (continued)
Each step is a separate HTTP request
We need to manage the data over a number of HTTP requests
The data is disparate (e.g., persons, pub type, etc.)
We need special Java objects that encapsulate the needed info

HTML form validation
At different steps different forms
Different forms for different publication types
A special login form
Apply form bean validation provided by Struts

User management
Login with a special login form
Logout
User information should be available for all administration actions
Store it in the session object
public final class User {
    ...  
    public void setPassword(String password) {
        if((password == null) || (password.length() == 0)) {
            throw new IllegalArgumentException(...);
        }
        password_ = password;
    }
    ...  }

Login Action

<action
    path="/login"
    type="edu.iicm.publication.struts.LoginAction"
    name="login_form"
    scope="request"
    validate="true"
    input="/login.jsp">
    <forward name="success" path="/manage.jsp"/>
    <forward name="failure" path="/login.jsp"/>
</action>

Login form bean

public final class LoginForm extends ActionForm {
    private String username_ = null;
    private String password_ = null;
    ...
    public String getPassword() {
        return password_;  
    }
    public void setPassword(String password) {
        password_ = password;
    }
    ...  }

Login form validation

public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    if((username_ == null) || (username_.length() == 0)) {
        errors.add("username",
            new ActionMessage("errors.username.required");
    }  
    return errors;
}  

User Database

Encapsulated as a Java object
Public interface provides a single method: checkUsername()
A particular implementation manages usernames and passwords
In an XML file, a database, memory, etc
Very simple solution in the example
Hardcoded in a hash table

```java
public final class UserDatabase {
    private HashMap users_;
    ...
    private void init() {
        users_ = new HashMap();
        users_.put("dhelic", "dhelic");
    }
    ...
}
```

We need very often a UserDatabase object
Each time a user wants to log in
We can use a shared single instance of the UserDatabase class
More efficient (e.g. created and loaded only once)
More secure because nobody can create another instance

```java
public final class UserDatabase {
    private static UserDatabase instance_ = null;
    ...
    public static UserDatabase getUserDatabase() {
        if(instance_ == null) {
            instance_ = new UserDatabase();
        }
        return instance_;
    }
    ...
}
```

How to enforce that nobody creates another instance
Declare the constructor as private

```java
public final class UserDatabase {
    ...
    private UserDatabase() {
        init();
    }
    ...
}
```

Login Action
Declare the constructor as private

```java
public final class LoginAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
        String username = ((LoginForm) form).getUsername();
        String password = ((LoginForm) form).getPassword();
        ActionMessages errors = new ActionMessages();
        if(!UserDatabase.getUserDatabase().checkUser(username, password)) {
            errors.add(ActionErrors.GLOBAL_MESSAGE,
            new ActionMessage("errors.authentication.failed"));
        }
        ...
    }
    ...
}
```

```java
String username = ((LoginForm) form).getUsername();
String password = ((LoginForm) form).getPassword();
ActionMessages errors = new ActionMessages();
if(!UserDatabase.getUserDatabase().checkUser(username, password)) {
    errors.add(ActionErrors.GLOBAL_MESSAGE,
    new ActionMessage("errors.authentication.failed"));
}
...
```
What is BaseAction?
It is always good to create an abstract BaseAction as a subclass of Struts action.
Declare an abstract execute() method:

```java
public abstract class BaseAction extends Action {
    ...
    public abstract ActionForward execute(ActionMapping mapping,
                                            ActionForm form,
                                            HttpServletRequest request,
                                            HttpServletResponse response);
    ...
}
```

Advantages of an abstract base action:
- You can implement a number of common methods here.
- You can use these methods within all subclasses.
- User management as example:
  - All management actions need to check if user is authenticated.

```java
public abstract class BaseAction extends Action {
    protected boolean isUserAuthenticated(HttpServletRequest request) {
        HttpSession session = request.getSession();
        ...
        User user = (User) session.getAttribute(Constants.USER_KEY);
        if(user == null) {
            return false;
        }
        return true;
    }
}
```

Source code:
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/struts/BaseAction.java

Logout Action:
```xml
<action
    path="/logout"
    type="edu.iicm.publication.struts.LogoutAction">
    <forward name="success" path="/manage.jsp"/>
</action>
```

The action removes the user object and invalidates the session:
```java
public final class LogoutAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                  HttpServletRequest request, HttpServletResponse response) {
        HttpSession session = request.getSession();
        session.removeAttribute(Constants.USER_KEY);
        session.invalidate();
        return mapping.findForward(Constants.SUCCESS);
    }
}
```

Example:
http://coronet.iicm.edu/struts-publicationdb
Example Zip:
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb.zip
Example WAR:
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb.war

Ruby on Rails:
Ruby is a pure object-oriented programming language.
- Pure means everything is an object, e.g. the number 1 is an instance of class Fixnum
- Interpreted scripting language
- Dynamically, weakly typed
- Single inheritance, but can be extended with so-called modules (similar to Java interfaces)
- Rich text processing functionality (similar to Perl)
- Rails is open source Web application framework
- http://www.rubyonrails.org/
- Supports development of database-backed application
- User-oriented Web database applications
- Follows MVC architecture and design pattern
Three main guiding principles

- Model-driven (domain-driven) development
  - You start with a data model and add the functionality, controllers, views on top of it
- Convention over configuration
  - Set of naming conventions (similar to JavaBeans but more in depth)
  - Less software, i.e., less code
  - Generating default code that you adjust to fit your needs

Domain-driven development

- Based on an ORM framework called ActiveRecord
- [http://wiki.rubyonrails.com/rails/pages/ActiveRecord](http://wiki.rubyonrails.com/rails/pages/ActiveRecord)
- ActiveRecord is a generic ORM framework
  - Similar to Hibernate
  - Uses a naming convention to provide the default mapping
  - You can adjust the default mapping if you need to

ActiveRecord naming convention

- [http://ar.rubyonrails.com/classes/ActiveRecord/Base.html](http://ar.rubyonrails.com/classes/ActiveRecord/Base.html)
- Names of classes and tables
  - Give names to your classes as English singular, and to your tables as English plural
  - Start the name of the class with an upper case, all other letters are lower case (Student)
  - Table name is all lower case (students)

Names of the table columns and instance variables

- Map 1-to-1, i.e., student.name maps to name column in students
- Primary key must be named id in the table
- Immediately you can use all methods from ActiveRecord

```ruby
@students = Student.find_all
@student = Student.new
```

Internally, ActiveRecord uses a single table to map the whole class hierarchy

- Single Table Inheritance [http://wiki.rubyonrails.com/rails/pages/SingleTableInheritance](http://wiki.rubyonrails.com/rails/pages/SingleTableInheritance)
- To map associations a simple domain language-like set of macros is used
  - For example, belongs_to, has_many, etc.
  - [ActiveRecord::Associations module](http://api.rubyonrails.com/classes/ActiveRecord/Associations/ClassMethods.html)

```ruby
class Student < ActiveRecord::Base
  has_and_belongs_to_many :courses
end
```

Connects two classes via an associative table (many-to-many relation)

- The name of the table: courses_students
- The names of foreign keys: course_id, student_id

Convention over configuration

- A lot of examples in ActiveRecord
- Further examples in controller module of MVC
- Controllers are modules that handle user requests
- Convention on mapping of URLs onto methods in controllers
  - Much easier than Struts configuration
URL = /controller_class_name/controller_method_name
For example, http://localhost:3000/test/index
Another example: http://localhost:3000/test/hello
class TestController < ApplicationController
def index
  render_text "Wow, that was easy"
end
def hello
  render_text "Hello World"
end
class StudentController < ActiveRecord::Base
scaffold :student
end
Whenever you have a model class, e.g., Student class you can use a so-called CRUD scaffold
create, read, update, delete methods
These methods are provided by the ActiveRecord

This single line embeds all of the CRUD methods into controller
Consequently, they are immediately visible through URLs
Read: http://localhost:3000/student/show/1
Update: http://localhost:3000/student/edit/1
List: http://localhost:3000/student/list
Note how meaningful and consistent URLs are (we discussed this in the first lecture)

Courses and students registered for courses
Two entities: Course and Student
Many-to-many relation between Course and Student
  One student might register for a number of courses
  For each course there is a number of registered students

create table courses ( id int not null auto_increment, title varchar(200), url varchar(200), description text, primary key(id) );
create table students ( id int not null auto_increment, name varchar(80), study_field varchar(10), primary key(id) );
create table courses_students ( course_id int not null, student_id int not null, primary key(course_id, student_id) );
By scaffolding you also get default views
However, you can adjust them
For a particular controller method, e.g., show() you need to create
show.rhtml
Another naming convention

First step with Rails: Create database (MySQL)
Follow the naming convention
create table courses ( id int not null auto_increment, title varchar(200), url varchar(200), description text, primary key(id) );
create table students ( id int not null auto_increment, name varchar(80), study_field varchar(10), primary key(id) );
create table courses_students ( course_id int not null, student_id int not null, primary key(course_id, student_id) );
Second step with Rails: Generate app skeleton
Invoke: rails courses
You will get a directory structure:
app/
config/
db/
doc/
lib/
log/
public/
script/
test/
vendor/
Within app directory you get something like:

app/
  controllers/
  helpers/
  models/
  views/

Edit config/database.yml
Start the app, invoke: ruby script/server

Third step: generate model and controller
Invoke: ruby script/generate model Course
Invoke: ruby script/generate model Student
Invoke: ruby script/generate controller Course
Invoke: ruby script/generate controller Student

Use the scaffold (the generated default controller and views)
class CourseController < ApplicationController
  scaffold :course
end
class StudentController < ApplicationController
  scaffold :student
end
Fourth step: Work (!) and adjust the app!

Adjusting the app
Create alternative views
Update the method in controller, e.g. the show method
Create the show.rhtml in views and adjust it

Implement the association
class Student < ActiveRecord::Base
  has_and_belongs_to_many :course
end

Adjust edit.rhtml and update method in controller
<% for @course in @courses %>
  <% if @student.courses.include? @course %>
    <%= @course.title %> :
    <input type="checkbox" name="<%= @course.title %>">
  <% else %>
    <%= @course.title %> :
    <input type="checkbox" name="<%= @course.title %>">
  <% end %>
<% end %>
def update
  @courses = Course.find_all
  @student = Student.find(@params[:id])
  if @student.update_attributes(params[:student])
    @student.courses.clear
    for course in @courses
      if @params[course.title]
        @student.courses<<(course)
    end
  end
  ...
Example: http://coronet.iicm.edu/mmis2/examples/mvc/rails/courses.zip
Ruby on Rails: Advances Features

- Defining layouts (headers, footers, ...)
- Modules for standard functionality, e.g., authentication
- Caching
- Validation and callbacks
- Transactions
- Testing

Further Readings (1/4)

- MVC Wiki with MVC History
  http://c2.com/cgi/wiki?ModelViewController
- J2EE Patterns: MVC
- Introduction to Jakarta Struts Framework from onjava.com
- Learning Jakarta Struts, part 2 from onjava.com

Further Readings (2/4)

- Learning Jakarta Struts, part 3 from onjava.com
- Struts articles from JSP and Servlets from onjava.com
  http://www.onjava.com/topics/java/JSP_Servlets
- Struts Best Practices from javaworld.com
- Struts Tutorial

Further Readings (3/4)

- Rails Wiki
  http://wiki.rubyonrails.com/rails
- Four Days on Rails
  http://www.rails4days.pwp.blueyonder.co.uk/Rails4Days.pdf
- Fast-track your Web apps with Ruby on Rails

Further Readings (4/4)

- Rails Wiki
  http://wiki.rubyonrails.com/rails
- Four Days on Rails
  http://www.rails4days.pwp.blueyonder.co.uk/Rails4Days.pdf
- Fast-track your Web apps with Ruby on Rails

Data Management in Web Applications

Denis Helic

Data Management - Classification of Data (1/2)

- Classifying data according to its structure
- Structured data
  - Clear separation between content and structure
  - Each data item has same structure
  - E.g., registered students, publications, etc.

Data Management - Classification of Data (2/2)

- Semi-structured data
  - Separation between content and structure
  - Structure can differ from a data item to a data item
  - E.g., documents (XHTML, SVG, PDF, ...)
  - E.g., a paragraph can include only text, but also links, lists, tables, etc...
- Managing of semi-structured data - Native XML databases
Structured Data Management (1/2)

Managing of structured data → DBMS, especially RDBMS

ACID databases test

Atomicity (transactions, either all or none)
Consistency (data integrity)
Isolation (transactions independent on each other)
Durability (permanent storage)

Structured Data Management (2/2)

Declarative query language SQL

You tell the system what you want not how to do it!
Less errors!

Maturity of the technology
Mature products, know-how, experience, support, etc.

Architecture of User-oriented Database applications

Problem with OO systems and RDBMS (1/8)

Process logic module (Model) has two submodules (OO technology)
Application logic (behaviour) of the system
Methods you can invoke on the objects
Highly dynamic, objects sending messages to each other
Data part
RDBMS has only data
Tables, fields, relations between tables

Problem with OO systems and RDBMS (2/8)

Difference between OO paradigm and relational paradigm
OO paradigm models the real world as a dynamic entity
Object relations (inheritance, composition, etc.) model the real world
Through the object interactions this entity changes its state frequently
Relational paradigm models a single state of the real world using only data
With external (declarative) languages you can alter this state

Problem with OO systems and RDBMS (3/8)

The problem is how to persist objects, their relations, and dynamics within tables
E.g., how to persist inheritance?
E.g., when to persist (each time when something happens in the OO system)?
It is a question of mapping OO systems to RDBMS
This problem is known as Impedance Mismatch

Problem with OO systems and RDBMS (4/8)

Simple example from Agile Database Techniques by Scott W. Ambler
We have an order and each order has a number of order items
Order has attributes such as dates, taxes, etc.
Order item has the number of ordered items
Behaviour: you can cancel order, get total taxes, etc.

Problem with OO systems and RDBMS (5/8)

Declerative query language SQL
You tell the system what do you want not how to do it!
Less errors!
Maturity of the technology
Mature products, know-how, experience, support, etc.
The ideal solution is to have one-to-one mapping of classes onto tables. In this simple case this is possible. In a general case this is not always possible. The second step is to map class attributes onto table fields. dateFulfilled from the class onto DateFulfilled from the table.

However, even in this simple case there are differences. Several tax attributes in the class, a single field in the table. Calculation is needed to implement this mapping. Primary keys, foreign keys in tables, references in objects. To implement the mapping correctly objects need to know about keys. Different data types. Currency type compared to float type.

But why model it in this way? Why do not we model it so that there is no such differences? Legacy systems, there is already a database schema, and you are building an OO system around that schema. If the system is new then we can take more care. The best way to build the OO schema first. Let the OO schema drive the implementation of the database schema. The OO model is the model that your application works with!

The mapping consists of three steps. Mapping attributes onto columns. An attribute maps onto zero or more columns in a table. Mapping classes onto tables. Handling of inheritance hierarchies. Mapping of associations, aggregation (composition) by relating tables to each other.

Four possible ways to map a class hierarchy. Using one table for an entire class hierarchy. Using one table per subclass. Using one table per class. Using one table per superclass and one table for subclasses.

Advantages
- Very simple
- Changing person roles is very simple
- Query performance is good (everything is in one table)

Disadvantages
- Adding a new attribute anywhere in the hierarchy → modifying the table
- A lot of empty fields in the table
- One person can only have a single type
### Mapping objects onto relational databases

#### Using one table per subclass

<table>
<thead>
<tr>
<th>Student</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>tid</td>
</tr>
<tr>
<td>&lt;&lt;primary key&gt;&gt;</td>
<td>&lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>mat_number</td>
<td>salary</td>
</tr>
</tbody>
</table>

**Advantages**
- Simple
- Query performance is good (everything is in one table)

**Disadvantages**
- Adding a new attribute to the superclass → modifying all the tables
- Person changes roles → copying the data between the tables
- Person can have multiple roles, but hard to maintain integrity

#### Using one table per class

<table>
<thead>
<tr>
<th>Person</th>
<th>Student</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>peid</td>
<td>peid</td>
<td>peid</td>
</tr>
<tr>
<td>&lt;&lt;primary key&gt;&gt;</td>
<td>&lt;&lt;primary key&gt;&gt;</td>
<td>&lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
<td>salary</td>
</tr>
<tr>
<td>mat_number</td>
<td>&lt;&lt;foreign key&gt;&gt;</td>
<td>value</td>
</tr>
</tbody>
</table>

**Advantages**
- Conforms to object-oriented concepts (the data stored in the tables that correspond to the types)
- Easy to extend the class hierarchy

**Disadvantages**
- Query performance is not that good (a lot of joins)
- You need a lot of SQLs to implement different views on the data

#### Using one table per superclass and one table for subclasses

<table>
<thead>
<tr>
<th>Person</th>
<th>PersonAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>peid</td>
<td>peid</td>
</tr>
<tr>
<td>&lt;&lt;primary key&gt;&gt;</td>
<td>&lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td>type</td>
<td>&lt;&lt;foreign key&gt;&gt;</td>
</tr>
<tr>
<td>name</td>
<td>key</td>
</tr>
<tr>
<td>value</td>
<td></td>
</tr>
</tbody>
</table>

**Advantages**
- Simple
- Query performance is OK (the data is in two tables)

**Disadvantages**
- A lot of data type conversion (different attributes are all mapped onto a single data type)
- One person can only have a single type (move this into another table, etc.)

### Mapping aggregation (composition)

- Implemented by relating tables to each other with primary and foreign keys
- Aggregation (composition) represents whole/part relationship
  - Composition a bit stronger (components belong to a single owner)
  - Aggregation allows that components are shared
- Associations just relate one object to another

- Mapping of associations and aggregation (composition)
- At database implementation level
- Everything that you do to the whole you always do to the parts
  - E.g., load, update, delete, etc.
- Mapping associations
  - It is different from a case to a case
Mapping objects onto relational databases

Mapping of many-to-many associations

E.g., a teacher may work in a number of schools
E.g., a school has a number of teachers

<table>
<thead>
<tr>
<th>Teacher</th>
<th>m</th>
<th>School</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>- salary : int</td>
<td></td>
<td>- address : String</td>
<td></td>
</tr>
</tbody>
</table>

The table in the middle is usually called the associative table.

O/R Mapping publications example

Managing of publications

Each publication has
- One or more persons involved (authors or editors)
- Title
- Year of publishing
- Optionally URL
- Type (depends on format)

The Model

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/

Publication superclass
- Holds 4-5 attributes that are common for all publication types
- All publication types are subclasses of the superclass (14 types)
- Each publication type has approximately 3-4 additional attributes

Which alternative for mapping class hierarchies to choose?

Alternative 1: a single table
- We will have a lot of fields (approx. 4+14*3)
- For each record the most of the fields would be empty

Alternative 2: a table per subclass
- 14 tables is too much for such a simple model

Alternative 3: a table per class
- 15 tables is too much for such a simple model

Alternative 4: a table for superclass and a single table for all subclasses
- 2 tables are OK
- The superclass table holds the common attributes
- The subclasses table holds key-value pairs according to the type
- Seems reasonable, because no space is wasted

O/R Mapping publications example

BibTeX format http://www.din1505.informationskompetenz.net/

Types such as Article, Conference, Book, etc

Depending on type different additional attributes
- Article has a journal
- Book has a publisher, etc.

Publication table

create table publication ( pid int not null auto_increment, title varchar(200) not null, url varchar(200), bibtype enum("Article", ...) not null, year varchar(4) not null, primary key(pid) );
Publication attributes table
create table pubattr ( pid int not null, name varchar(30) not null, value varchar(200) not null, primary key(pid, name) );

We have person table
create table person ( peid int not null auto_increment, name varchar(80), primary key(peid) );

Association between publication and persons is a many-to-many
A publication can have many authors
An author can work on a number of publications
Thus, we need to map this association using an associative table
create table pubperson ( peid int not null, pid int not null, role enum("Author", "Editor"), primary key(peid, pid) );

We need to map this association using an associative table.

Complete SQL
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/setupdb.sql

Interface between process logic and data management
Programmatic (API) → no SQL in there
Another very important rule
No SQL in the process logic
Otherwise mixture of process logic and data management
SQL only allowed through the implementation in the data management module

In all cases we need JDBC
http://java.sun.com/products/jdbc/
However, we want to abstract JDBC and SQL from process logic
The implementation needs to hide it from process logic
Alternative 1: Implement the abstraction yourself
Using design patterns + JDBC

Alternative 2: Use an existing implementation and declare the mapping only
E.g., Java Data Objects from Sun
http://java.sun.com/products/jdo/
E.g., Enterprise Java Beans
http://java.sun.com/products/ejb/
O/R Mapping framework: Hibernate
http://www.hibernate.org/
SQL statement problems
JDBC API provides the Statement class
Statement objects usually applied in the following way
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery(
"select * from publication where title regexp " +
request.getParameter("title");

Usually long statements → typing errors
Security considerations
Suppose that users type ‘something ; select * from passwords;’
Performance issues
Each time the statement is compiled again!

Solution: use PreparedStatement class
select_pubs = connection.prepareStatement("select * from publication where title regexp ?");
select_pubs.setString(1, request.getParameter("title");

No typing errors
No need to compose the query string
Types are checked, e.g. setString(), setInt()
The security issue resolved
Parameter is used as a value of the query variable
Performance improved
PreparedStatement is pre-compiled only once

Authentication and connection pooling
DB authentication usually implemented in the following way
Connection connection = DriverManager.getConnection(
"jdbc:mysql://localhost/publicationdb",
"username", "password");

A simple solution
Declare username and password as init-param in web.xml
Another problem of the previous approach
Performance since each request opens a new connection
A better solution
Resolve authentication issue together with connection pooling

Connection pools
Broker class encapsulates access to the database connections with get-Connection() method
Behind the scene broker manages a buffer of connections
goingConnection() returns the first available connection
If no free connection enlarge the buffer
When a client finishes it frees the connection (broker notified)
Optimizing the buffer size!

Usually JDBC drivers provide connection pooling
Apache Tomcat provides connection pooling
Database Connection Pool (DBCP)
Part of Jakarta Commons project
http://jakarta.apache.org/commons
Implementing the abstraction - JDBC issues(9/14)

DBCP uses Java Naming Directory Interface (JNDI)
- JNDI Data Source
- Define JNDI resource reference in web.xml
- Map JNDI resource onto a real resource in server.xml
- Lookup JNDI data source in the code

Implementing the abstraction - JDBC issues(10/14)

Define JNDI resource reference in web.xml

```xml
<resource-ref>
  <res-ref-name>jdbc/publicationdb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Implementing the abstraction - JDBC issues(11/14)

Map JNDI resource onto a real resource in server.xml

```xml
<Context path="/struts-publicationdb"
docBase="struts-publicationdb"
debug="0" reloadable="true" >
  <ResourceParams name="jdbc/publicationdb">
    <parameter>
      <name>username</name>
      <value>publicationdb</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>mmis2004</value>
    </parameter>
  </ResourceParams>
</Context>
```

Implementing the abstraction - JDBC issues(12/14)

Map JNDI resource onto a real resource (continued)

```xml
<Parameter>
  <name>driverClassName</name>
  <value>org.gjt.mm.mysql.Driver</value>
</Parameter>
<Parameter>
  <name>url</name>
  <value>jdbc:mysql://localhost/publicationdb</value>
</Parameter>
```

Implementing the abstraction - JDBC issues(13/14)

Lookup JNDI data source in the code

```java
Context init = new InitialContext();
Context ctx = (Context) init.lookup("java:comp/env");
DataSource ds = (DataSource) ctx.lookup("jdbc/publicationdb");
connection_ = ds.getConnection();
```

Implementing the abstraction - JDBC issues(14/14)

Be careful!
- Need to notify the broker when finished
- Do so by closing connection and all other DB resources

```java
try {
  ....
} finally {
  close(insert_person, result);
  closeConnection();
}
```

Design Patterns - Data Access Object (DAO)(1/5)

Encapsulates all database access into a single class
- CRUD interface (create, read, update, delete)
- Works with transfer objects
- TOs reflect the model

Design Patterns - Data Access Object (DAO)(2/5)

Example of DAO CRUD interface
- PersonDAO
  ```java
  public void storePerson(Person person);
  ....
  public Iterator readAllPersons();
  public Person readPersonWithId(int id);
  ....
  ```
- Source code
  http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/db/PersonDAO.java
Design Patterns - Data Access Object (DAO)(3/5)

- Similar classes for Publication
  - [http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationDAO.java](http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationDAO.java)
- Code to access the DAOs and present the results (Search interface)
  - PublicationDAO dao = DAOFactory.createPublicationDAO();
  - ... 
  - StringBuffer buffer = new StringBuffer();
  - Iterator pubs = dao.readAllPubs(type, year, title);
  - while (pubs.hasNext()) {
    - Publication pub = (Publication) pubs.next();
    - buffer.append(pub.getStringRep(writer));
  - }

Design Patterns - Data Access Object (DAO)(4/5)

- To decouple the storage from DAO class work with interfaces
- Work with Abstract Factory Pattern to obtain proper DAO instances
- Allows you to move to another storage, e.g., XML native database

Design Patterns - Data Access Object (DAO)(5/5)

- Two levels of implementation
  - An abstract class that implements the DAO interface
  - It declares a number of abstract methods to access the storage
  - Source code
    - [http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationDAOImpl.java](http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationDAOImpl.java)
  - A particular implementation implements the abstract methods
  - Source code
    - [http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationJDBCDAOImpl.java](http://coronet.iicm.edu/mmis2/examples/mvc.struts-publication/src/edu/iicm/publication/db/PublicationJDBCDAOImpl.java)

Using a mapping framework - Hibernate

- Hibernate is an open source Java O/R mapping framework
  - [http://www.hibernate.org](http://www.hibernate.org)
- Hibernate allows you to simple declare the O/R mapping
  - The implementation is already included in the framework
- Currently widely used (the most popular O/R tool)
- It supports all major RDBMS

Technical infrastructure needed by Hibernate(1/12)

- You develop the Model as Java classes, objects, etc.
- For this OO model you declare a mapping onto relations
  - Recollect the suggestion that the OO model should drive the mapping
  - The objects and classes from the Model need only to follow common Java
    - Java bean idiom

Technical infrastructure needed by Hibernate(2/12)

- Java beans specification
- Very simple specification
  - For each class attribute you need getAttrName() and setAttrName()
  - For each boolean class attribute you need isAttrName()
  - You need an empty public constructor
  - Basically, it is a naming convention

Technical infrastructure needed by Hibernate(3/12)

- How the Java Beans specification can be useful?
  - E.g., Struts form beans
  - Login form bean
    - `<form-bean
      name="login_form"
      type="edu.iicm.publication.struts.LoginForm" />

Technical infrastructure needed by Hibernate(4/12)

- Login form bean
  - public final class LoginForm extends ActionForm {
    - private String username_ = null;
    - private String password_ = null;
    - ...
    - public String getPassword() {
      return password_;
    }
    - public void setPassword(String password) {
      password_ = password;
    }
    - ...
  - }
Struts reads the parameters submitted with the HTML form. The names of parameters are used to create method names. E.g., setPassword(), setUsername(). Struts invokes these methods on the instance of the form bean class. How Struts can invoke these methods in run-time? The invocations are not hard-coded!

Java Reflection API
http://java.sun.com/docs/books/tutorial/reflect/
This API allows you to investigate run-time properties of Java classes and objects. E.g., what is the name of the superclass of this class? E.g., give me the list of public methods of this class? E.g., give me an instance of a method of an object!!! Method is a class that can have instances.

Additionally, it allows you to instantiate objects, invoke methods, etc. E.g., give me an instance of this Method, call the invoke() method on that instance. It is equivalent to a hard-coded method invocation. In this way Struts can call methods on the instance of the form bean class.

Example of Java reflection
Publication attributes table
create table pubattr (
pid int not null,
name varchar(30) not null,
value varchar(200) not null,
primary key(pid, name)
);

Depending on the publication type we have different additional attributes. Article type http://coronet.iicm.edu/mmis2/examples/mvc/struts-publication/src/edu/iicm/publication/Article.java In the pubattr table we have key-value pairs. E.g., (journal, "JUCS") The key of this pair should be used to obtain the name of the method. E.g., getJournal(), setJournal().

Using Java reflection we can then invoke the proper method and set the attribute.

// prepare arguments for reflection
key = key.substring(0, 1).toUpperCase() +
key.substring(1, key.length());
Object[] args = new Object[1];
args[0] = value;

try {
    Method setter = pub.getClass().getMethod("set" +
    key, class_args);
setter.invoke(pub, args);
} catch (SecurityException exc) {
...
}

The same approach is used by Hibernate. You need to specify what attributes of your Java objects you want to persist. Hibernate uses these names to invoke the proper getters and setters. E.g., for Publication class you specify "title" attribute. Hibernate will generate getTitle() and setTitle() and invoke them with reflection API. Additionally, you define how to map those attributes onto relations. Hibernate hides the JDBC and SQL issues from you.

Publication attributes table
create table pubattr (pid int not null, name varchar(30) not null, value varchar(200) not null, primary key(pid, name));

In the pubattr table we have key-value pairs. E.g., (journal, "JUCS") The key of this pair should be used to obtain the name of the method. E.g., getJournal(), setJournal().

Using Java reflection we can then invoke the proper method and set the attribute.

final protected void populatePubAttributes(Publication pub, ...
HashMap attributes) {
    ...
    // use Reflection API to invoke setters
    Class[] class_args = new Class[1];
class_args[0] = String.class;

    Iterator attrs = attributes.keySet().iterator();
    while (attrs.hasNext()) {
        String key = (String) attrs.next();
        String value = (String) attributes.get(key);
        ...
        try { ...]

// prepare arguments for reflection
key = key.substring(0, 1).toUpperCase() +
key.substring(1, key.length());
Object[] args = new Object[1];
args[0] = value;

try {
    Method setter = pub.getClass().getMethod("set" +
    key, class_args);
setter.invoke(pub, args);
} catch (SecurityException exc) {
...
}
Model
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication

All concerned classes are Java beans
We have inheritance (Publication and subclasses)
We have a many-to-many relation (Publication and Person)
For each class that we want to make persistent we need a declaration file

Person.hbm.xml
<hibernate-mapping>
<class name="edu.iicm.publication.Person" table="person">
  <id name="id" column="peid">
    <generator class="increment"/>
  </id>
  <property name="name" column="name"/>
</class>
</hibernate-mapping>

Publication.hbm.xml
<hibernate-mapping>
<class name="edu.iicm.publication.Publication" table="publication">
  <id name="id" column="pid">
    <generator class="increment"/>
  </id>
  <property name="title" column="title"/>
  <property name="year" column="year"/>
  <property name="url" column="url"/>
</class>
<set name="persons" table="pubperson">
  <key column="pid"/>
  <many-to-many column="peid" class="edu.iicm.publication.Person"/>
</set>
</hibernate-mapping>

Publication.discriminator
<discriminator column="bibtype" type="string"/>
<subclass name="edu.iicm.publication.Article" discriminator-value="Article"/>
<subclass name="edu.iicm.publication.Unpublished" discriminator-value="Unpublished"/>
</class>

PersonHIBERNATEDAO

static {
  try {
    session_factory_ = new Configuration().addClass(Person.class).buildSessionFactory();
  } catch (Exception exc) {
    log_.error("Initial SessionFactory creation failed.", exc);
    throw new ExceptionInInitializerError(exc);
  }
}

HibernateDAO
Actually, we do not need this since Hibernate already abstracts JDBC and SQL
With DAO we can abstract Hibernate from the process logic ;)
Additionally, Hibernate configuration is accomplished by DAO
Belongs to the Data Management Module

The mapping files should accompany class files
Hibernate resolves automatically associations, inheritance, etc.
e.g., proper subclasses will be created depending on the value of the "bibtype" field
Persons will be loaded whenever a Publication is loaded
You can also configure at what time the Persons should be loaded
e.g., eager load → immediately
e.g., lazy load → when a person is requested

Source
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/Person.hbm.xml
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/Publication.hbm.xml
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/Publication.discriminator
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/PersonHIBERNATEDAO
public void storePerson(Person person) {
    Session session = session_factory_.openSession();
    Transaction transaction = session.beginTransaction();
    session.save(person);
    transaction.commit();
    session.close();
}

public Iterator readAllPersons() {
    Session session = session_factory_.openSession();
    String query_string = "from edu.iicm.publication.Person" + "in class edu.iicm.publication.Person" + "where person.id=:id";
    Query query = session.createQuery(query_string);
    query.setInteger("id", id);
    return (Person) query.uniqueResult();
}

public Person readPersonWithId(int id) {
    Session session = session_factory_.openSession();
    String query_string = "select person from person " + "in class edu.iicm.publication.Person " + "where person.id=:id";
    Query query = session.createQuery(query_string);
    query.setInteger("id", id);
    return (Person) query.uniqueResult();
}

Similar for Publication DAO

Source code PublicationHIBERNATEDAO

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PublicationHIBERNATEDAOImpl.java

Stand-alone demonstration

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/test/PublicationDBTest.java

Hibernate uses HQL: The Hibernate Query Language

HQL Manual


Very similar to SQL

But is fully OO

You search in classes, and object collections, not in relations

Core J2EE Patterns - Data Access Object


Hibernate Your Data

http://www.onjava.com/pub/a/onjava/2004/01/14/hibernate.html

Hibernate Reference Manual


Get started with Hibernate


Struts Recipes: Hibernate and Struts


Web Services

Denis Helic
One of the basic principles of Software Engineering - Separation of Concerns
We separate different aspects of the system into different modules
Each module supports a single part of the functionality
The modules interact with each other to provide the needed functionality
How do we connect modules, i.e., how they interact with each other?

The communication through interfaces
We can observe interfaces from different view points
From the implementation point of view
Programmatic (procedural) interface (API)
Declarative interfaces, e.g., XML, SQL, etc.
Hybrid, i.e., API + Declarative

Another way of observing interfaces is the runtime environment
All modules in one OS process (modules = objects)
Modules distributed in a number of processes on the same machine (modules = components)
Modules distributed over the network

Finally, we can observe interface concerning the way of coupling of modules
Tightly coupled
Loosely coupled
Decoupled (if not connected to each other)

If a modification to one module leads to a modification of another module than tightly coupled
Otherwise loosely coupled
If operation of a module depends on operation of another module than tightly coupled
Otherwise loosely coupled
If decoupled the modules are totally independent
They can still communicate through a middleware module
Service-Oriented Architecture (9/15)

Usually, APIs are closer to tight coupling
- e.g., a modification to DAO interface requires a modification in the process logic
  - Without the modification the system can not function anymore
- Typically, declarative interfaces are closer to loose coupling
  - e.g., adding a new XML element to an XML document processed by an XSLT
  - The system can function but the new functionality requires an extended XSLT

Service-Oriented Architecture (11/15)

Service Oriented Architecture (SOA) is an architectural style
- The goal of this style is to achieve loose coupling among services interacting via declarative interfaces over the network
- A service is a piece of functionality
  - e.g., DAO service provides CRUD interface to the underlying database

Service-Oriented Architecture (13/15)

Descriptive (declarative) messages constrained by a schema
- The semantics is in messages
- The schema prescribes the structure and the vocabulary of messages
  - e.g., header, body of the message
- No (or minimal) system behaviour is included in the messages
- The schema is extensible
  - No extension breaks the previous versions of services

Service-Oriented Architecture (15/15)

Request - response cycle of the DAO service
- Request message containing “Person” and the name of the person
- The DAO service accesses the database and retrieves the data
- Internally, it creates a Person object
- Response message contains a description of this object

Web Services and SOA (1/7)

Web Services are a specific application of SOA
- Obviously, the network infrastructure is the Web
- Other important features of SOA are also mapped onto the Web by Web Services
  - e.g., using of HTTP, XML, etc.
- Web Services standardization by Web Consortium
  http://www.w3.org/2002/ws/
How Web Services comply with the SOA constraints

Generic communication interfaces
- Interfaces based on Internet and Web protocols such as HTTP, SMTP, FTP, etc.
- e.g., HTTP request (GET, POST)
- e.g., HTTP response
- e.g., HTTP messages with HTTP header, HTTP body, etc.
- e.g., no specification what to do or how to do something!

Messages are encoded as XML
- e.g., no predefined semantics (each application defines its own semantics)
- e.g., well-formedness of XML
- e.g., XML schemas for XML validity
- e.g., schemas are extensible

Where is the message structure
- i.e., message header, message body?
- Defined by a particular style of Web Services
- SOAP Web services
  - SOAP = Simple Object Access Protocol

In addition to SOAP, Web services introduce WSDL
- WSDL = Web Services Description Language
- WSDL is also developed by the Web Consortium

Sample Web service(1/4)
- A Web service is a piece of functionality
- It is offered for public by a service provider
- We have a server where we host the service
- I.e., the service has its URL
- e.g. http://localhost:8080/axis/services/EchoService

Sample Web service(2/4)
- Simple example of a service
- The service prints out the string it receives (echo)

```
public class EchoService {
    public String echo(String msg) {
        return msg;
    }
}
```

Source code
http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/ws/EchoService.java

e.g., HTTP GET

Invoking a Web service using encoded URL over HTTP GET

Encoded URL will be transformed into a SOAP message

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
<SOAP-ENV:Body>
<echo>
  <msg>Hello World!</msg>
</echo>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Very simple encoding
- `echo` element is the method
- `msg` subelement is the argument

The Web service guesses the proper datatype conversions
- In this case very simple because the argument and the return value are simple strings

Web Services Frameworks (1/2)

Frameworks provide abstractions of SOAP and WSDL
- Basically, they allow you to do RPC transparently
- You do not even need to know about SOAP or WSDL
- .NET from Microsoft
  - http://www.microsoft.com/net/
- WebSphere from IBM
- J2EE from Sun
  - http://java.sun.com/developer/technicalArticles/J2EE/j2ee_ws/

Web Services Frameworks (2/2)

- Apache Axis
  - http://ws.apache.org/axis/
  - Open source Java-based Web services framework
  - Started as a SOAP framework
  - Evolved to support WSDL, etc.
  - Download, the latest version is 1.2 RC2
  - Installation guide

Web Services Description Language - WSDL (1/5)

WSDL is a schema language for Web services
- WSDL document describes
  - Datatypes that a Web service uses (e.g., PurhcaseOrder schema)
  - Operations that a Web service provides
  - The location of a Web service, i.e., its URL

Web Services Description Language - WSDL (2/5)

WSDL of the EchoService
```xml
<wsdl:binding name="EchoServiceSoapBinding" type="impl:EchoService">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="echo">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="echoRequest">
    </wsdl:input>
    <wsdl:output name="echoResponse">
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Web Services Description Language - WSDL (3/5)

```xml
<wsdl:message name="echoResponse">
<wsdl:part name="echoReturn" type="soapenc:string"/>
</wsdl:message>
<wsdl:message name="echoRequest">
<wsdl:part name="msg" type="soapenc:string"/>
</wsdl:message>
```

Automatically generated WSDL for EchoService

Web Services Description Language - WSDL (4/5)

WSDL is a structured description of a Web service
- It is written in a machine-understandable way
  - e.g., a client accessing the WSDL of a service can invoke the service as described
- Automatization of the process of Web services invocation
Automatization of creation of clients
- Automatically generate clients from WSDL descriptions
- Abstract the SOAP/WSDL completely and hide it behind an API
- Automatization of creating services
- Use your existing functionality
- Create automatically WSDL

Java2WSDL tool creates automatically WSDL from Java interfaces
- It creates descriptions of all public methods
- It creates proper datatypes
- It needs to be able to access these Java classes
- The classes should be JavaBeans

SOAP and WSDL in Axis

WSDL2Java tool creates Java classes from WSDL descriptions
- Client side
  - A Java interface which corresponds to the operations defined in WSDL
    - For echo service this would be Echo interface with a single public method
  - A so-called stub, which is the implementation of the generated interface
    - It uses Axis API (e.g., Service, Call) and SOAP to communicate with the Web service
  - A ServiceLocator class which retrieves instances of the generated interface
    - Basically it is an Abstract Factory

Server side
- An empty implementation of the generated interface
- You should modify this empty implementation
- Forward the calls to the proper methods of your implementation

SOAP and WSDL in Axis - An Example

PersonDAO interface from the Publication Database
- Basically, we want to offer the access to the database through a Web service
  - The usual way of providing Web services
    - I.e., publishing already existing functionality as Web services

Using Java2WSDL
org.apache.axis.wsdl.Java2WSDL
-o persondao.wsdl
-l "http://coronet2.iicm.edu/axis/services/persondao"
-n urn:coronet.iicm.edu/persondao
-p "edu.iicm.publication.db" coronet.iicm.edu/persondao
edu.iicm.publication.db.PersonDAO

Source code
http://coronet.iicm.edu/mmis2/examples/ws/persondao.wsdl

Using WSDL2Java
org.apache.axis.wsd1.WSDL2Java
-o src
-d Session
-s
-p edu.iicm.publication.db.ws
persondao.wsdl
Client side

PersonDAO interface for the client

```java
package edu.iicm.publication.db.ws;

public interface PersonDAO extends java.rmi.Remote {
    public void storePerson(edu.iicm.publication.db.ws.Person in0)
        throws java.rmi.RemoteException;
    public java.lang.Object readAllPersons()
        throws java.rmi.RemoteException;
    public edu.iicm.publication.db.ws.Person readPersonWithId(int in0)
        throws java.rmi.RemoteException;
}
```

Source code

[http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/PersonDAO.java](http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/PersonDAO.java)

---

Person class created from WSDL

```java
package edu.iicm.publication.db.ws;

public class Person implements java.io.Serializable {
    private int id;
    
    public int getId() {
        return id;
    }
    
    public void setId(int id) {
        this.id = id;
    }
    
    // other methods...
}
```

Source code

[http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/Person.java](http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/Person.java)

---

Service side

```java
PersonDAOImplementation

... edu.iicm.publication.Person person =
    dao.readPersonWithId(in0);
    edu.iicm.publication.db.ws.Person out0 =
        new edu.iicm.publication.db.ws.Person();
    out0.setId(person.getId());
    out0.setName(person.getName());
    out0.setRole(person.getRole());
    return out0;
}
```

Source code

[http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/PersonDaoSoapBindingImpl.java](http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/PersonDaoSoapBindingImpl.java)

---

Client code

```java
PersonDAOServiceLocator locator = new PersonDAOServiceLocator();

try {
    PersonDAO dao = locator.getPersonDAO();
    Person person = new Person();
    person.setName(name);
    dao.storePerson(person);
    log_.debug("Person stored");
}
```

Source code

[http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/client/PersonDAOClient.java](http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/client/PersonDAOClient.java)

Example with Eclipse

SOAP Monitor


HTTP GET

Problems of SOAP/WSDL Protocol Stack (1/2)

- The basic problem: it is exclusively used for Remote Procedure Calls
- RPC implies an API
- Recollect that APIs tend to enforce tight coupling of modules and systems
- It is somewhat confusing
- We use declarative XML to describe APIs
- Regardless if you use Java interfaces or XML it is still an API

Problems of SOAP/WSDL Protocol Stack (2/2)

- APIs rely on a close world assumption
- You have modules and systems that are tightly coupled
- You need to know all of your users, their systems, tools
- You need to enforce a particular API
- Everyone needs to know about the changes in the API
- Scalability problem!

World Wide Web: Open World (1/5)

- But what do we have on the Web
  - Obviously it is not a closed world
  - Impossible to know all of your users
  - Impossible to enforce API or changes in API
  - The only possibility is to have a very simple basic and extensible framework

World Wide Web: Open World (2/5)

- But we already have that
  - A standardized, extensible application protocol: HTTP
  - A standardized, extensible message format: HTML
  - A single unifying namespace of URLs
  - Allows to create links that make the Web actually
  - URLs keep the Web together as a single huge application

World Wide Web: Open World (3/5)

- Now, let us extend a notion of URLs (URIs)
  - URLs identify resources: IICM, Homepage of IICM, ...
  - Representations of resources are delivered over the Web by means of HTTP
    - Most notably, HTML representations
  - But other representations are possible, such as GIFs, JPEGs, etc.

World Wide Web: Open World (4/5)

- But we are interesting in declarative representations
  - E.g., XML-based representations
  - That particular extension is the basis for resource-oriented Web Services
  - HTTP, URIs, XML
  - It is extremely loosely coupled (as the Web itself)
  - We do not care what is on the other side (a file, a method, a module, etc.)
  - As long as we can use simple HTTP GET + URI and get XML out of it

World Wide Web: Open World (5/5)

- Note here the difference between RPC and this new approach
  - To get an XML representation out of a RPC service you need to know
    - Its URL
    - Its WSDL
  - Then you need to build a SOAP message accordingly

Some advantages of Resource-oriented Web Services

- Extensibility
  - You organize a web service around URLs
  - That service can automatically integrate with other web services through links
  - Similar to how HTML pages are linked together
- Performance
  - Anything that is retrieved with HTTP GET and URL can be cached
  - Caching is already built in into HTTP
  - These two advantages were the reasons for the success of the Web
Representational State Transfer: REST(1/4)

- A particular architectural style for Web Services
- Based on the premises from above
  - HTTP, URLs, XML
  - Use HTTP to its full extent
    - GET, POST, PUT, DELETE
  - Manage and manipulate resources on the Web
    - Concentrate on data (XML) rather than on software (RPC)

Representational State Transfer: REST(2/4)

- How to manipulate resources
- CRUD interface
  - Create = PUT or POST
  - Read = GET
  - Update = POST or PUT
  - Delete = DELETE
- We have everything!
  - Just need to create XML vocabularies to represent resources

Representational State Transfer: REST(3/4)

- Comparison with SOAP/WSDL
  - The majority of SOAP/WSDL based services
    - Simply provides CRUD interface
    - Why then take SOAP/WSDL and add one level of complexity
  - Now we can think of a Web server as a huge information repository, a database
  - The Web is a huge distributed hypermedia information repository that links all Web servers into a single system

Representational State Transfer: REST(4/4)

- Everything is already there
  - Everything has been there for almost ten years
  - No need to wait for standards, implementations, etc.
  - Just use HTTP, URLs, XML to:
    - Publish information on the Web
    - Retrieve information from the Web
    - Link information on the Web

How to create a REST Web service(1/6)

- To create a REST Web service the following steps should be accomplished
  - Define URLs
  - Define XML vocabulary
  - Define which methods are available at each URL
  - Define HTTP status codes

How to create a REST Web service(2/6)

- Defining URLs
  - Recollect the clean URLs that we have discussed before
    - Course with ID 1: http://localhost:3000/courses/1
    - Course with ID 2: http://localhost:3000/courses/2
    - All courses: http://localhost:3000/courses/
  - Note that URLs are regular, i.e., we can automatically construct them

How to create a REST Web service(3/6)

- Defining XML formats
  - Single course
    <course>
      <title>MMIS2</title>
      <url>http://coronet.iicm.edu/lectures/mmis2</url>
      <id type="integer">1</id>
      <description>Advanced course on multimedia information systems</description>
    </course>

How to create a REST Web service(4/6)

- All courses (collection)
  <courses>
    <course>
      <title>MMIS2</title>
      <url>http://coronet.iicm.edu/lectures/mmis2</url>
      <id type="integer">1</id>
      <description>Advanced course on multimedia information systems</description>
    </course>
    ...</course>
  </courses>
Defining methods

**http://localhost:3000/courses/** GET and POST

- GET retrieves the list, POST adds another course to the list

**http://localhost:3000/courses/1** GET, PUT, DELETE

- PUT updates the course

- Additionally, send OPTIONS at each URL to see what is available

---

### REST Web service: Example(1/9)

- We need a Web server that can dispatch URLs and HTTP methods to appropriate code
- Within that code we manipulate resources using their XML representations
- With Cocoon we can easily dispatch URLs (recollect sitemap)
- With Rails we can also dispatch URLs easily
- Recollect that Rails dispatch to a class and a method within that class
- We need to dispatch to a class and according to the HTTP method

---

### REST Web service: Example(2/9)

- There is a Rails plugin that does exactly that


  ```ruby
  class CourseController < ApplicationController
  include RestController::Base
  ...
  end
  ```

---

### REST Web service: Example(3/9)

- The plugin allows you to introduce resources and within resources handle HTTP methods

  ```ruby
  resource :collection do |r|
  conditions << @courses = Course.find_all
  r.post do
  ...
  end
  end
  
  GET method is automatically handled and connected with appropriate XML-based view
  ```

---

### REST Web service: Example(4/9)

- collection.rxml in views directory

  ```ruby
  xml.instruct!
  xml.courses do
  xml.title 'Courses'
  @courses.each do |course|
  xml.course do
  xml.id course.id
  xml.title course.title
  xml.url course.url
  xml.description course.description
  ...
  end
  end
  ```

---

### REST Web service: Example(5/9)

- Handling POST to the collection (adding a course)

  ```ruby
  r.post do
  @course = Course.new params[:course]
  if @course.save
  render_post_success :action => 'by_id', :id => @course
  end
  end
  ```

---

### REST Web service: Example(6/9)

- Handling of single courses (introduce a new rule by_id)

  ```ruby
  resource :by_id do |r|
  conditions << @course = Course.find(params[:id])
  r.put do
  @course.attributes = params[:course]
  if @course.save
  render_put_success
  end
  end
  ```
Deleting a resource

```
r.delete do
  if @course.destroy
    render_delete_success :id => nil
  end
end
```

GET XML view

```
xm.instruct!
@course.to_xml
```

Demo:

```
http://coronet.iicm.edu/mmis2/examples/ws/rest/rest_courses.zip
curl http://localhost:3000/courses
curl http://localhost:3000/courses/1
curl -i -X POST --data-charset=UTF-8 -d '<course><title>Web Information Systems</title><description>Very advanced course</description>' http://localhost:3000/courses

curl -i -X PUT -H 'Content-Type: application/xml' -d '<course><title>Web Information Systems</title><description>Very advanced course</description>' http://localhost:3000/courses/4
curl -i -X DELETE http://localhost:3000/courses/4
```

Ajax and Web 2.0

The intent is to make web pages feel more responsive

- By exchanging small amounts of data with the server behind the scenes
- The entire web page does not have to be reloaded each time the user makes a change
- Meant to increase the web page's interactivity, speed, and usability

A possibility to build Rich Web Clients

- Javascript widget libraries
  - Rico: http://openrico.org/
  - Dojo: http://dojotoolkit.org/
  - Echo2: http://www.nextapp.com/platform/echo2/echo/demo/

It can be very powerful when combined with REST services

I hope to see some demonstrations during your presentations ;-)
Further Readings (1/3)

What is Service-Oriented Architecture?
A Web Services Primer
Top Ten FAQs for Web Services
http://webservices.xml.com/pub/a/ws/2002/02/12/webservicefaqs.html?page=1
Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools
Articles on http://webservices.xml.com/ from O’Reilly

Further Readings (2/3)

SOAP Primer from W3C
http://www.w3.org/TR/2003/REC-soap12-part0-20030624/
SOAP Tutorial
http://www.w3schools.com/soap/default.asp
Java and XML: SOAP
http://www.onjava.com/pub/a/onjava/excerpt/java_xml_2rch2/
WSDL Tutorial
http://www.w3schools.com/wsdl/default.asp
Apache Axis User Guide
http://us.apache.org/axis/java/user-guide.html

Further Readings (3/3)

REST definition (PhD Thesis of Roy Fielding)
The Restful Web: Column on xml.com
http://xml.com/pub/at/34
Putting REST on Rails
http://www.xml.com/pub/a/2006/04/19/rest-on-rails.html
REST vs SOAP: Second Generation Web Services
http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html

Information Discovery (1/3)

Web today: more than 10 billions of Web pages
How to discover information on the Web (very hard!)?
- Browsing
- Searching
Very often results are disappointing!

Information Discovery (2/3)

How do we find things in real world?
Example 1: The Library
- You want to find a book on Relational Databases
- Use the computer in the library to list books by author, title, subject, etc.
- Identify the book you want and look in the list where the book is
Both examples based on metadata

Information Discovery (3/3)

Example 2: The Video Store
- You want to find the latest Star Wars movie
- Use the computer in the store to list books by director, title, actors, etc.
- Identify the video you want and look in the list where the video is

Metadata (1/4)

Metadata is data about data
Example 1: Data about books
- Author: Nick Scerbakov
- Subject: Databases
- Title: From Databases to Hypermedia
- ...
Example 2: Data about videos
- Director: George Lucas
- Title: Star Wars - Revenge of the Sith
- Actor: Ewan McGregor
- ...

Metadata comes in key-value pairs
- Key: Author
- Value: Nick Scerbakov

For different applications we use different sets of metadata
- Library: author, title, subject, ...
- Video store: director, title, actors, ...

We can use metadata for other purposes as well!
- Cataloging
- Printing
- Sorting, etc.

Use metadata to describe Web resources
- Web sites, Web pages, etc.

We can apply metadata on the Web in the same way as in the real world
- searching, cataloging, printing, etc.

We can apply metadata in a more Web-specific way
- provide a summary of a Web site, describe intellectual property rights of a Web page, etc.

Current situation with metadata on the Web
- Not that much metadata on the Web
- Especially, not that much in HTML pages
- <meta> element in HTML
  - used to raise search engine rankings!
  - search engines do not query <meta> elements directly

Some systems apply metadata for more than 10 years now! (e.g. Hyperwave)

What are the challenges for metadata on the Web?
- Web comprises many different applications
  - Many libraries on the Web
  - Many video stores on the Web
- ...

Applications use different sets of metadata, but the principles are the same (i.e. metadata comes always in key-value pairs)

What do we need to use metadata effectively on the Web?
- Technology:
  - That supports the basic metadata principle
  - That allows to use different sets of metadata
  - That is compatible with current Web standards
What do we need to use metadata effectively on the Web (continued)?

Standard (standardized technology):
- We want to exchange metadata with other people
- We want software to process it automatically
- We need to agree on a standard for metadata!

Resource Description Framework (RDF) is such standardized technology
Developed by Web Consortium (Recommendation)
http://www.w3.org/RDF
Part of larger initiative called Semantic Web
http://www.w3.org/2001/sw

Resource Description Framework (RDF)
Resource Description Framework is a recent development of Web Consortium
http://www.w3.org/RDF
Purpose of RDF is to provide a standard for exchanging metadata on the Web
The basic RDF specification consists of:
- RDF Data Model: http://www.w3.org/TR/rdf-concepts - status Recommendation

Semantic Web - Introduction
New initiative by Web Consortium
Definition (from W3C): The Semantic Web is the representation of data on the World Wide Web
Description of Web resources in the form of metadata

Design ideas behind Semantic Web
- Based on the current Web technologies, such as URL (URI) and XML
- Extension of the current Web
- Make it very simple to create metadata! (recollect the reason for the success of the Web)
- Make it possible to introduce different metadata sets
- Anyone, anywhere, anytime can provide metadata for a resource by addressing it with its URI (similar to linking in HTML)

Goals of Semantic Web
- Web of descriptions and resources - Semantic Web
- Enable effective information discovery (e.g. search)
- Enable automation (e.g. software agents)
- Enable effective integration (e.g. ftp and http)

Example: Web resource described with some metadata
Web resource http://coronet.iicm.edu/lectures/mmis2
Creator: Denis Helic
Administrator: Denis Helic
Title: Multimedia Information Systems
...
RDF Data Model is based on the following rules:

- A **Resource** is anything that can have a URI (e.g., all Web pages, all Web images, all files accessible through FTP, etc.)

- A **Property** has a name and describes some relationship (e.g., Creator, Title, Subject, etc.)

- A **Statement** consists of a combination of a Resource, a Property and a value

  "The creator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic"

Statement may be represented as a triple (N3 notation):

<http://coronet.iicm.edu/lectures/mmis2> <creator> <Denis Helic>.

A particular description of a resource includes a number of statements

"The creator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic"

"The administrator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic"

"The title of http://coronet.iicm.edu/lectures/mmis2 is Multimedia Information Systems 2"

A value might be a resource

"The homepage of http://coronet.iicm.edu/lectures/mmis2 is http://coronet.iicm.edu"

"The homepage of http://coronet.iicm.edu/lectures/mmis2 is http://coronet.iicm.edu"

RDF graph consists of nodes and arcs

- Oval nodes represent resources
- Square nodes represent values
- Arcs (arrows) represent properties

RDF Data Model

Graphically

A home page might be a resource

"The homepage of http://coronet.iicm.edu/lectures/mmis2 is http://coronet.iicm.edu"
Advanced concepts in RDF

- A Property is also a Resource
- A Statement is also a Resource
- Both can have their own properties (e.g. when it was created?)

How this model meets the design ideas?

- To match the basic metadata principle - each triple is a combination of a URI and a key-value pair
- A simple data model - just triples
- Scalable - easy to handle even large number of triples
- Anyone can provide metadata for any resource - if you know its URI!
- Independent - no prescribed sets of metadata
- Interchangeable - we need to specify XML syntax for it!

RDF/XML Syntax Specification

- Defines how to encode an RDF graph into a valid XML
- We need to represent nodes and arcs as XML elements, attributes, element content and attribute values
- A graph is a collection of paths of the form Node, Arc, Node, ...
- In RDF/XML these turn into sequences of nested elements which alternate between elements for Nodes and Arcs

Example 1

First path: creator

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
</rdf:Description>
```

Second path: administrator

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <administrator>Denis Helic</administrator>
</rdf:Description>
```

Third path: title

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <title>Multimedia Information Systems 2</title>
</rdf:Description>
```

Example 1 (complete)

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
  <administrator>Denis Helic</administrator>
  <title>Multimedia Information Systems 2</title>
</rdf:Description>
```

Example 2

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
  <administrator>Denis Helic</administrator>
  <title>Multimedia Information Systems 2</title>
</rdf:Description>
```

Example 1:
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf

Example 2:
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf
Example 2 (continued)

<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
<creator>Denis Helic</creator>
<administrator>Denis Helic</administrator>
<title>Multimedia Information Systems 2</title>
<homepage>
  <rdf:Description rdf:about = "http://courses.iicm.edu">
    <creator>Hermann Mauerer</creator>
    <administrator>Karl Trummer</administrator>
    <title>Courses offered by IICM</title>
  </rdf:Description>
</homepage>
</rdf:Description>

To complete the example we need declaration!

<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns = "http://coronet.iicm.edu/mmis/examples/rdf#">
  ...
</rdf:RDF>

Example 2:
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_complete.rdf

striping pattern!

To represent collections we use rdf:Bag, rdf:Seq or rdf:Alt

<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <topics>
    <rdf:Bag>
      <topic>Internet</topic>
      <topic>XSLT</topic>
      <topic>RDF</topic>
      <topic>Cocoon</topic>
    </rdf:Bag>
  </topics>
</rdf:Description>

query language for RDF

SELECT ?x
WHERE (<http://somewhere/res1>, <http://somewhere/pred1>, ?x)

rdf:Bag is unordered collection
rdf:Seq is sequence of properties
rdf:Alt is set of alternate properties (e.g. represent title in different languages)

To represent collections we use rdf:Bag, rdf:Seq or rdf:Alt

<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <topics>
    <rdf:Bag>
      <topic>Internet</topic>
      <topic>XSLT</topic>
      <topic>RDF</topic>
      <topic>Cocoon</topic>
    </rdf:Bag>
  </topics>
</rdf:Description>

Query language for RDF
SELECT ?x
WHERE (<http://somewhere/res1>, <http://somewhere/pred1>, ?x)

RDFS. Part of Jena RDF framework:
http://jena.sourceforge.net/

More on the Jena framework in the next lecture
Anyone can provide metadata for any resource - if you know its URI!
The first description: 
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf 
An additional description: 
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_add.rdf

<rdf:Description
rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
...
</rdf:Description>
Refer to resource with the URI!

An aggregator tool collects all metadata about a resource
We can query all metadata about the specific resource
Metadata might be distributed all over the Web, i.e., we have a distributed network of metadata
Basics of Semantic Web!

Anybody can define metadata sets
RDF Schema: defines all properties
Example Dublin Core Metadata Schema (Library background)

<table>
<thead>
<tr>
<th>Content</th>
<th>Intellectual Property</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Creator</td>
<td>Date</td>
</tr>
<tr>
<td>Subject</td>
<td>Publisher</td>
<td>Type</td>
</tr>
<tr>
<td>Description</td>
<td>Contributor</td>
<td>Format</td>
</tr>
<tr>
<td>Language</td>
<td>Rights</td>
<td>Identifier</td>
</tr>
<tr>
<td>Relation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dublin Core: http://dublincore.org/
In many cases you need just Dublin Core
RDF Schema: http://dublincore.org/2003/03/24/dces#

Expressing bibliographies in RDF
Allows you to exhange bibliographic information more easily
PublicationDB
http://coronet.iicm.edu/struts-publicationdb
Publications
http://coronet.iicm.edu/denis/pubs/list.rdf

Further examples on http://www.rdfdata.org/
Open directory RDF dump
http://rdf.dmoz.org/
US State and Regions
http://www.daml.ri.cmu.edu/ont/USRegionState.daml
AI bibliography
http://www.isi.edu/webscenario/planning.scheduling.daml

FOAF Project
 Friend of a friend
Creating RDF descriptions of people, contacts, homepages, etc.
To support communication, networking between people on the Web
URL http://www.foaf-project.org/

Expressing bibliographies in RDF
Allows you to exhange bibliographic information more easily
PublicationDB
http://coronet.iicm.edu/struts-publicationdb
Publications
http://coronet.iicm.edu/denis/pubs/list.rdf
Applications using RDF and other Semantic Web technologies

Easiest way to implement Semantic Web applications

Using a Semantic Web framework

Jena Framework

Open source, Java-based, implemented by Hewlett Packard

Project homepage

http://jena.sourceforge.net/

Jena framework provides

A programmatic environment for RDF and other Semantic Web technologies

RDF API, i.e., manipulating RDF graphs using an API

Reading and writing RDF in RDF/XML, N3 and N-Triples

In-memory and persistent storage (with RDBMS)

RDQL - a query language for RDF

API for other Semantic Web technologies

Publishing RDF graphs on the (Semantic) Web

Build a server-side application for manipulating of RDF graphs (models)

Basic RDF Server (BRDFS)

Implemented as a master thesis at IICM

Implemented using Jena Framework

I.e., Jena framework accessible through Web interface

Requirements for BRDFS

Running as a Web application, i.e., available through HTTP

Persistent data

Can create, delete, update models and statements

Can merge models, query models, etc.

Easy to use for humans and software agents

BRDFS implementation

As a Web application using Tomcat

Server is an action framework, i.e., similar to Struts

You can issue commands to the server using the action framework

HTML user interface

BRDFS Run-time

At start time the server is connected to a MySQL database

After that every RDF model operation is persisted in that database

Jena provides DB Connection

Jena abstracts the database layer

We just deal with models

Source code

http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/
edu/iicm/brdfs/action/NEWMODEL.java

Delete a model

ModelRDB m =
ModelRDB.open(
jena2_dbconnection_.getJena2DBConnection(),
modelname);
m.remove();
m.close();

Model name comes from the request

Source code

http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/
edu/iicm/brdfs/action/DELETEMODEL.java
Publishing RDF graphs on the (Semantic) Web (7/10)

Merge two models

ModelMaker maker = ModelFactory.createModelRDBMaker(jena2_dbconnection_.getJena2DBConnection());
Model target = maker.openModel(modelname);
target.begin();
...
Model merge_model = maker.openModel(tomerge_values[i]);
target.add(merge_model, true);
merge_model.close();
target.commit();
target.close();

Source code
http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/edu/iicm/brdfs/action/MERGEMODELS.java

Publishing RDF graphs on the (Semantic) Web (8/10)

Add statements

ModelMaker maker = ModelFactory.createModelRDBMaker(jena2_dbconnection_.getJena2DBConnection());
Model model = maker.openModel(modelname);
model.begin();
Resource resource = model.createResource(subject);
Property property = model.createProperty(namespace, propertyname);
resource.addProperty(property, object);
model.commit();
model.close();

Source code
http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/edu/iicm/brdfs/action/ADDSTATEMENTS.java

Publishing RDF graphs on the (Semantic) Web (9/10)

Query a model

ModelRDB model = ModelRDB.open(jena2_dbconnection_.getJena2DBConnection(), modelname);
Query query = new Query(querystring);
query.setSource(model);
QueryExecution qe = new QueryEngine(query);
QueryResults results = qe.exec();
response_writer_.writeQueryResult(results);
results.close();

Source code
http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/edu/iicm/brdfs/action/QUERYMODEL.java

Publishing RDF graphs on the (Semantic) Web (10/10)

RDQL is a query language for querying RDF models

RDQL Tutorial
http://jena.sourceforge.net/tutorial/RDQL/index.html
Similar to SQL
But operations on triples not on relations
SELECT ?x
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith")

BRDFS Simple Example (1/3)

BRDFS accessible from
http://coronet.iicm.edu/brdfs
We upload two MMIS2 course RDF files from the last lecture
Intellectual property rights of the MMIS2 homepage
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf
Information about topics of the course
http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_add.rdf
We merge the models and query the final model
Shows also how metadata integration is achieved

BRDFS Simple Example (2/3)

Get creator of the MMIS2 homepage
SELECT ?creator
WHERE (<http://coronet.iicm.edu/lectures/mmis2> <http://coronet.iicm.edu/mmis2/examples/rdf#creator> ?creator)

BRDFS Simple Example (3/3)

Get the course topics
SELECT ?topic
WHERE (<http://coronet.iicm.edu/lectures/mmis2> <http://coronet.iicm.edu/mmis2/examples/rdf#topic> ?topic)

BRDFS Publication Database (1/10)

Another master thesis project at IICM
Implement OAI provider for the JUCS journal
JUCS journal
http://www.jucs.org/
You can retrieve BibTeX reference for each of the articles
Open Archives Initiative - OAI
http://www.openarchives.org/
Providing metadata about publications in digital libraries
Metadata format is not specified
Should support at least Dublin Core format
Defines also a number of protocols for accessing and retrieving the data

OAI provider for JUCS built on the top of BRDFS
Metadata managed in RDF
Different output formats available
BibTeX format encoded as RDF, i.e., as XML
Dublin Core format encoded as RDF, i.e., as XML

Automatic adding of references for new issues of the journal
Automatic conversion of BibTeX text format onto BibTeX RDF
Adding of statements to an already existing RDF model
HTML format for people
Supports browsing
Automatic conversion of XML formats using XSLTs on the client side

Administration interface
http://coronet.iicm.edu/brdfs
Uploading of new BibTeX reference files
Retrieving of the model
Querying of the model

Select IDs and titles of all articles
SELECT ?id, ?title
WHERE (?id <http://coronet.iicm.edu/bibtex#journal> "Journal of Universal Computer Science")
(?id <http://coronet.iicm.edu/bibtex#title> ?title)

Select titles and authors
SELECT ?title, ?author
WHERE (?x <http://coronet.iicm.edu/bibtex#journal> "Journal of Universal Computer Science")
(?x <http://coronet.iicm.edu/bibtex#title> ?title)
(?x <http://coronet.iicm.edu/bibtex#author> ?author)
USING bibtex FOR <http://coronet.iicm.edu/bibtex#>

Select titles and authors for specific issues
SELECT ?title, ?author
WHERE (?x bibtex:journal "Journal of Universal Computer Science")
(?x bibtex:title ?title)
(?x bibtex:author ?author)
USING bibtex FOR <http://coronet.iicm.edu/bibtex#> AND ?volume == 1
Add another issue to the database
- Shows the integration of metadata
- HTML user interface
  http://coronet.iicm.edu/brdfs/oai/
- With XSLT on the client side

---

RDFS - RDF Schema(1/9)

- RDF is very simple
  - It provides a basic set for information modelling
  - Because of this simplicity it is a perfect assembly language
    - It is possible to build other information modelling languages on top of it
  - RDF Schema is one such language
    http://www.w3.org/TR/rdf-schema/
  - RDFS is developed by the Web consortium

---

RDFS - RDF Schema(2/9)

- RDF Schema is an object-oriented modelling language
  - It allows you to define classes
  - It allows you to define relationships between classes
    - E.g., subclass relationship
  - It allows you to define properties of classes

---

RDFS - RDF Schema(3/9)

- How does RDFS relate to RDF?
  - Classes are resources
    - Each class has a URL, so we can define properties for it
  - If two classes are related with subClassOf property then we build class hierarchies
  - Simple properties of a class are same as properties that have simple values in RDF

---

RDFS - RDF Schema(4/9)

- RDFS Example
  - At the top of the class hierarchy we have WaterSource
  - Two subclasses of WaterSource are Stream and BodyOfWater
    - Stream has subclasses such as River, Creek, etc.
    - BodyOfWater has subclasses such as Lake, Sea, etc.
  - There is a relationship between Stream and BodyOfWater
    - The relationship is called emptiesInto

---

RDFS - RDF Schema(5/9)

- Definition of classes and relationships in RDFS
  - `<rdfs:Class rdf:ID="WaterSource"/>`
  - `<rdfs:Class rdf:ID="Stream">`
    - `<rdfs:subClassOf rdf:resource="#WaterSource"/>`
  - `<rdfs:Class rdf:ID="BodyOfWater">`
    - `<rdfs:subClassOf rdf:resource="#WaterSource"/>`
  - `<rdfs:Class rdf:ID="River">`
    - `<rdfs:subClassOf rdf:resource="#Stream"/>`
  - `<rdfs:Class rdf:ID="Lake">`
    - `<rdfs:subClassOf rdf:resource="#BodyOfWater"/>`
  - `<rdfs:Class rdf:ID="BodyOfWater">`
    - `<rdfs:subClassOf rdf:resource="#BodyOfWater"/>`
  - `<rdfs:SubClassOf rdf:resource="#WaterSource"/>`
  - `<rdfs:SubClassOf rdf:resource="#Stream"/>`
  - `<rdfs:SubClassOf rdf:resource="#BodyOfWater"/>`
  - `<rdfs:SubClassOf rdf:resource="#River"/>`
  - `<rdfs:SubClassOf rdf:resource="#Lake"/>`

---

RDFS - RDF Schema(6/9)

- Definition of classes and relationships in RDFS (continued)
  - `<rdfs:Property rdf:ID="emptiesInto"/>`
  - `<rdfs:range rdf:resource="#BodyOfWater"/>`
  - `<rdfs:domain rdf:resource="#River"/>`

---

RDFS - RDF Schema(7/9)

- Using RDF schema with RDF
  - Basically, we describe new real Web resources according to the developed schema
    - `<emptiesInto rdf:resource="http://www.china.org/geography#EastChinaSea"/>`

---
What are the advantages of applying RDF Schema to this small RDF graph?

- Inference!
- We can infer new facts from already existing facts and classes and relationships defined in the RDF schema.
- River is a subclass of Stream → Yangtze is a stream.
- Stream is a subclass of WaterSource → Yangtze is a WaterSource.
- River emptiesInto BodyOfWater → EastChinaSea is a BodyOfWater.

The newly inferred facts can also be used to improve information discovery:
- e.g., you are searching for streams that empty into EastChinaSea.
- You get the info about Yangtze.

OWL is built on the top of RDFS, i.e., it extends RDFS.
- Work prior to OWL:
  - DAML (developed by DARPA)
    http://www.daml.org/
  - OIL
    http://www.ontoknowledge.org/oil/index.shtml
- OWL developed by the Web consortium (Recommendation)
  http://www.w3.org/2004/OWL/

OWL is an extension of RDFS:
- It allows you to define richer relationships than RDFS.
- Inferencing capabilities are more powerful.
- Consequently, you can improve information discovery even more.

Using OWL to define rich properties:
```xml
<owl:ObjectProperty rdf:ID="connectsTo">
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
  <rdfs:domain rdf:resource="#WaterSource"/>
  <rdfs:range rdf:resource="#WaterSource"/>
</owl:ObjectProperty>
```

The connectsTo relationship is symmetric:
- i.e., if A connectsTo B, then B connectsTo A.
```
<River rdf:ID="Yangtze"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns="http://www.geodesy.org/water/naturally-occurring#">
  <connectsTo>
    <River rdf:about="http://www.china.org/rivers#Wu"/>
  </connectsTo>
</River>
```

Give me all rivers that connectTo Wu → Yangtze.

Using OWL to define a transitive property:
```xml
<owl:ObjectProperty rdf:ID="containedIn">
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Sea"/>
  <rdfs:range rdf:resource="#BodyOfWater"/>
</owl:ObjectProperty>
```

A transitive property:
```
<Sea rdf:ID="EastChinaSea"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns="http://www.geodesy.org/water/naturally-occurring#">
  <containedIn>
    <Sea rdf:about="http://www.china.gov#ChinaSea"/>
  </containedIn>
</Sea>
```

i.e., if A is containedIn B and B is contained in C then A is contained in C.
Web Ontology Language - OWL(7/8)

<Sea rdf:about="http://www.china.gov#ChinaSea"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.geodesy.org/water/naturally-occurring#">
  <containedIn>
    <Ocean rdf:about="http://www.geodesy.org#PacificOcean"/>
  </containedIn>
</Sea>

EastChinaSea is containedIn PacificOcean
subClassOf is a transitive property

Web Ontology Language - OWL(8/8)

Other properties of relationships
- FunctionalProperty, there is at most one value for the property
- InverseProperty
  - contains
  - contains
- DatatypeProperty, the value of the property is a simple datatype
  - XML schema datatypes

OWL Example(1/6)
Using OWL to improve information discovery while browsing
- We have a typical directory of information
e.g., similar to Yahoo categories
- We can browse through categories and find the information we need

OWL Example(2/6)
We can model categories as OWL classes
- OWL classes are related with subClassOf relation
- subClassOf relation is transitive
e.g., Class A is a subClassOf class B
i.e., an instance of class A is also an instance of class B
e.g., if a document belongs to a subcategory it also belongs to its super-category

OWL Example(3/6)
We had a lot of learning resources on Computer Science
We wanted to categorize them and improve information discovery
Firstly, we created an ontology of Computer Science
e.g., categorization of computer science fields
e.g., information systems, algorithms, etc.
Finally, we added resources to the categories
When you open a category you see all resources including resources from subcategories

OWL Example(4/6)
"SuchAlgorithmus" is a subClassOf "Algorithmus"
"BinäreSuche" is an instance of "SuchAlgorithmus"
"BinäreSuche" is therefore an instance of "Algorithmus"
Additionally, a category can be a subclass of more than one superclass
Example
http://coronet.iicm.edu/navig

OWL Example(5/6)
Using the system to personalize the information discovery
We define a personal class for each user
What happens if we declare the “Personal” class as a superclass of some class in Computer Science

OWL Example(6/6)
e.g., Personal is a superclass of “Algorithmus"
All instances of “Algorithmus” are instances of “Personal”
We access these instances directly
i.e., we declare our interest in algorithms
A new instance in algorithms → we see it immediately
Example
http://coronet.iicm.edu/navig
RDF Tools/Links

- Protege Editor
  http://protege.stanford.edu/
- IsaViz, visual authoring tool for RDF
  http://www.w3.org/2001/11/IsaViz/
- Jena RDF Framework, Java parser, Java API, RDQL
  http://jena.sourceforge.net
- W3C RDF Validator
  http://www.w3.org/RDF/Validator/
- As editor: any XML editor, emacs, etc.

Further Readings

- RDF Tutorial
  http://www710.univ-lyon1.fr/~champin/rdf-tutorial/
- Intro to RDF
  http://www.dlib.org/dlib/may98/miller/05miller.html
- RDF Tutorial and examples
  http://www.dstc.edu.au/Research/Projects/rdf/
- Semantic Web articles from http://www.xml.com
  http://www.xml.com/semweb/
- What is RDF?
  http://www.xml.com/pub/a/2001/01/24/rdf.html
- Metadata articles from http://www.xml.com
  http://www.xml.com/metadata/