Electronic commerce

course outline

Marcin Skubiszewski

19th February 2003

A Overview

This course teaches technical foundations of electronic commerce. We begin with a brief overview of the architecture of computers, and with a description of the principles governing the Internet. We then introduce the World-Wide Web, and the technologies that are directly involved in building electronic commerce applications: the Java language, JSP (Java Servlet Pages), Jakarta Struts, and relational databases. As a practical exercice, we describe in detail a typical electronic commerce application: a web-based Internet store. During laboratory work associated with this course, students build significant fragments of the store.

Additionally, we discuss computer security and the market of Internet-related software (these subjects will only be discussed if time permits).

B Detailed outline

B.1 A brief introduction to computers and to the Internet

1.1. **Introduction to computer hardware**

- 1.1.1 **The notion of bit**
- 1.1.2 **How bits are used to represent various kinds data**
- 1.1.3 **Major components of a computer**
  - i. **Main memory**
  - ii. **Processor**
  - iii. **Mass storage: magnetic disks**

1.2. **How computers run programs**

- 1.2.1 **How programs are written and executed**
  - i. **Compiled code**
    Compiled code *is a representation of a program that can be directly executed by a computer. Compiled code is very hard to understand by a human.*
  - ii. **Source code and compilation**
    Source code *is a representation of a program in a programming language (e.g., Java). Source code is easy to write, understand and modify by a human. Source code cannot be directly executed by a computer; instead, it can be* compiled, *and the resulting compiled code can be executed.*

iii. **Pseudocode and virtual machines**
Pseudocode *is a representation of a program that can be executed with the help of another program, called* virtual machine. *Pseudocode is used in the execution of Java programs (including programs that are going to be written as part of laboratory work associated with this course).*

1.2.2 The parallel execution of several programs (*multithreading* and *multiprocessing*)

1.3. How the Internet works

1.3.1 How the Internet transmits reliable streams of characters

i. **How packets are forwarded to destination**
In this item, we describe briefly two fundamental components of the Internet: the Internet Protocol (IP) and the Domain Name Service (DNS).
- what is a packet and why we use packets
- addressing and naming of computers
- routing

ii. **How packets are used to send reliable streams of characters**
In this item, we describe briefly a fundamental component of the Internet, the Transmission Control Protocol (TCP).
- sequence numbers; error detection; retransmission
- TCP ports; how a client connects to a server

1.3.2 The World-Wide Web

i. **HTTP (the hypertext transmission protocol)**
- How web documents are identified; the notion of URL (uniform resource locator)
- Metadata about documents: type (text, image, sound etc.), expiration date, the language used etc.
- Cookies
  Cookies allow a web server to identify requests as coming from a given user.

ii. **HTML (the hypertext meta-language)**
- The syntax of HTML
- The structure of an HTML document: headers, tables, images, links, frames
- Using Cascading Stylesheets (CSS) to customize the appearance of an HTML document

iii. **How a webserver operates**
- How files are served as static web documents
- How programs are called to generate dynamic web documents
  – server-side includes, i.e., mixing static document fragments with program fragments

iv. **How a web browser operates**
- The execution of client-side programs in the browser (as opposed to server-side programs, executed in the webserver)
  – Javascript
  – Flash animations
- Accessibility and performance
  Accessibility *is the collective name given to the issues related to users whose access to the web is somehow restricted (e.g., blind users, or users with slow connections based on cell phones).*

B.2 Building electronic commerce applications

This part of the course is centered around the Internet store project.

2.1. **Introduction: presentation of the project**

2.1.1 **Requirements: what the project should accomplish**

This item is covered in the document describing the project, entitled The Internet Store Project.

2.1.2 **The architecture of the project**

We introduce the major elements that need to be developed as part of the project:

- *a* database schema—*a description of the organization of our business-related data,*
- business logic, *i.e., programs that perform operations related to the store's business (searching for articles, taking orders, registering new customers and new articles for sale, etc.),*
- *the web pages to be displayed to the customers (written in JSP—Java Servlet Pages),*
- *a description of the graphical appearance of our web pages (written as a* CSS*— cascading stylesheet),*

2.2. **The software components that we use in the project**

This item is covered in the document describing the project.

2.3. **Introduction to object-oriented programming and to the Java language**

2.3.1 **Modularity**

In this item, we explain the principles according to which a programming project should be partitioned into smaller parts (modules). Each module must do a limited number of conceptually simple things.

2.3.2 **Objects in Java**

An object *is a composite data structre representing a real-world object, fact or relationship. An object comes with a number of associated pieces of programming, called* methods. *Objects are Java's way to achieve modularity.*

2.3.3 **Assorted Java constructs**

In this item, we describe various Java constructs necessary for completing the project.

i. **Variable declarations**
   - **Programming style: how to name variables**

ii. **How and where Java stores objects**

iii. **Difference between object and non-object values**

iv. **Composite instructions**
   - **Conditionals and loops**

v. **Predefined types: non-object types, strings, tables**

2.4. **Organizing an application with Jakarta Struts**

2.4.1 **The MVC2 (model-view-controller) separation**

The **model** *represents our business logic. The* **view** *represents the way in which we create webpages. The* ***controller*** *organizes the application.*

2.4.2 **The view**

i. **Writing JSPs (Java Servlet Pages) to represent objets**

ii. **Writing JSPs to represent complete web pages**
  - **JSP inclusion**
  - **tiles in Jakarta Struts**

2.4.3 **The controller**

i. **Actions in Jakarta Struts**
ii. **Forms in Jakarta Struts**

2.4.4 **The model**

- **programming the business logic in Java**

2.5. Using a relational database to organize data

2.5.1 General organization of data

i. **The notion of *relation* (aka *table*)**
A relation is a set of records, called tuples, *that represent real-world objects or facts of a given kind, and that conform to a common format.*
For example, a table called **customers** *may contain, for every customer of a store, a tuple containing the customer's name, her address, and other relevant information.*

ii. **SQL queries concerning only one table**
SQL (simple query language) is the language used to retrieve data from (or to insert data into) database tables.

iii. **The notion of index**
An index *is an auxiliary data structure that makes it possible to rapidly retrieve tuples from a table, based on certain values stored in said tuples (for example, to retrieve a tuple representing a customer, based on the customer's name).*

2.5.2 Designing a database according to the entity-relationship model

Due to time constraints, the treatment of this item will be very brief.

The entity-relationship model teaches us how to organize information in a database: the database should contain a table for every category of real-world objects that we want to describe (these objects are called entities*), and for every kind of relationship between such objects.*

i. **The representation of entities; primary keys**
ii. **The representation of relationships; foreign keys**
iii. **SQL queries involving more than one table**

2.5.3 Consistency and transactions in databases

Due to time constraints, the treatment of this item will be very brief.

i. **Example consistency problems**
  - **Data inconsistencies due to concurrent execution**
  - **Inconsistent effect of half-failed complex operations**

ii. **Transactions**
The mechanisms used to solve consistency problems in databases are collectively called transactions.

iii. **Practical advice: how to use transactions in our project**

2.6. JDBC

JDBC is a library that allows Java programs to access relational databases. Once you understand relational databases, JDBC is simple to use, and therefore our course about JDBC will be short.

B.3 Advanced topics (optional part)

Due to time constraints, it is likely that the subject matter described below will not be tought.

3.1. **Security**

3.1.1 **Classification of attacks against computer systems**

i. **Non-technical attacks** (exploiting human stupidity or lack of attention)

ii. **Insider attacks**
A majority of successful and truly harmful attacks comes from employees of the organization being attacked. It is often very hard to defend against such attacks

iii. **Exploiting bugs in server software**
This is the easiest way to attack a computer, because adequate software is readily available and easy to use. Every server accessible from the Internet is targetted several times a day by attacks of this kind.

iv. **Viruses**

v. **Denials of service**
A denial of service *attack is less severe than other kinds of attacks: the purpose is just to temporarily disrupt the operation of the targetted computer, and not destroy or steal data (as is the case with most other attacks).*
Denial of services attacks are very hard to defend against, and for this reason they deserve being discussed.

3.1.2 **Making software secure**

i. **The management of privileges**
   - **The notion of user**
   - **User privileges**
   - **Sandboxing**
     Sandboxing consists in severly limiting what a given program can do (enclosing the program in a sandbox), so that the program cannot do harm. Sandboxing is used whenever a web browser executes a program downloaded from the Internet.

ii. **Fighting security bugs**
   - **Software audit; the example of OpenBSD**
   - **Suppressing unnecessary services**

3.1.3 **Cryptography**

i. **Goals to achieve**
   - **secrecy**
   - **authentication, integrity and non-repudiation**

ii. **Cryptographic techniques**
   - **Secret key encryption**
     This is the traditional kind of encryption, it has been used for many years by the military.
     – **separation between the algorithm and the key**
       In a modern cryptosystem we distinguish between the algorithm *(the general encryption method, expressed as a computer program) and the* key *(a randomly chosen sequence of bits used in the encryption process). The algorithm and the key are both necessary for a cryptosystem to work.*
       While in older cryptosystems everything was secret, in modern systems the algorithm is publicly known, and only the key is secret.
       We explain the surprising fact that it is better to use a publicly-known algorithm than a secret one.

- **Public key cryptography**
  Public key cryptography is a modern technique that allows you to communicate in a secure way with people whom you never met before, and with whom you have never communicated before.
  For example, you can use public key cryptography to place an order with a bank with which you never did business before, and be certain (i) *that you are indeed talking to the bank in question, not to an impersonator, and* (ii) *that the communication is kept secret.*
  - **The MD5 digest**
  - **The electronic signature**
  - **Certificates and public key infrastructure**

3.1.4 Example privacy-related problems

i. **SPAM (unsolicited commercial email) and E-mail address harvesting**
ii. **Involuntary e-mail receipts**
iii. **Hidden text in Microsoft Word**

3.2. The market of Internet-related software

The market of internet-related software is deeply influenced by two phenomena:

- *the quasi-monopoly held by Microsoft on several segments of the market*
- *the abundance of open-source software, i.e., software that anyone can modify or distribute for free and without asking anyone for permission, because the authors have abandoned their intellectual property rights.*

3.2.1 The Microsoft monopoly

i. **How the monopoly appeared**
   - **How MS-DOS appeared and how it evolved into MS Windows**
   - **The battle between Internet Explorer and Netscape Navigator**
ii. **The antitrust lawsuit**
iii. **Effects of the monopoly**
   - **Microsoft's *.NET* vs Sun Microsystem's *Java***
   - **Microsoft's Windows Media Player**

3.2.2 Open source software (also known as *free software*)

i. **Definition**
ii. **Open source licenses**
iii. **Importance**
   Open source software exists for almost all the tasks that are commonly performed using computers (including, of course, the operation of a web server). Its use is especially widespread in Internet servers.
iv. **Why open source software is abundant**
   Unlike commercial software developers, open source software developers receive no money and no direct benefit whatsoever in exchange for their software. Therefore, the abundance of open source (free) software is astonishing: at first sight, free software should not be any more abundant than free lunches. We describe the strengths of open source software, that lead to its abundance.
v. **Organization: who writes and distributes open source software, and why**