Chapter 9: Application Design and Development

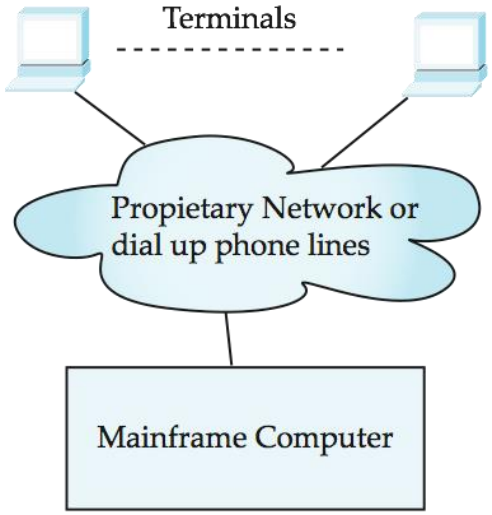
Database System Concepts

Chapter 9: Application Design and Development

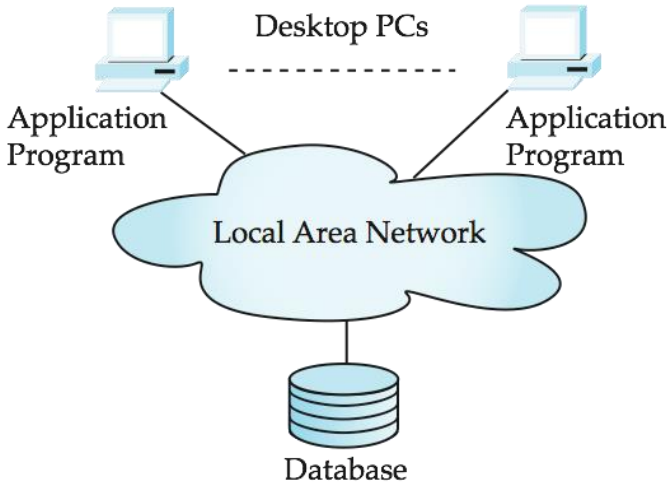
- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Application Architecture Evolution

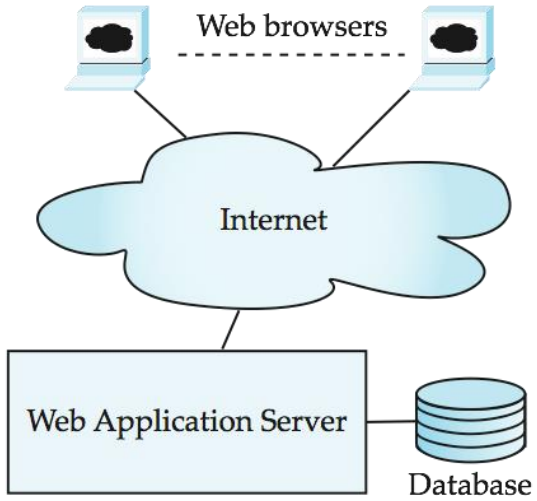
- Three distinct era's of application architecture
  - mainframe (1960's and 70's)
  - personal computer era (1980's)
  - Web era (1990's onwards)



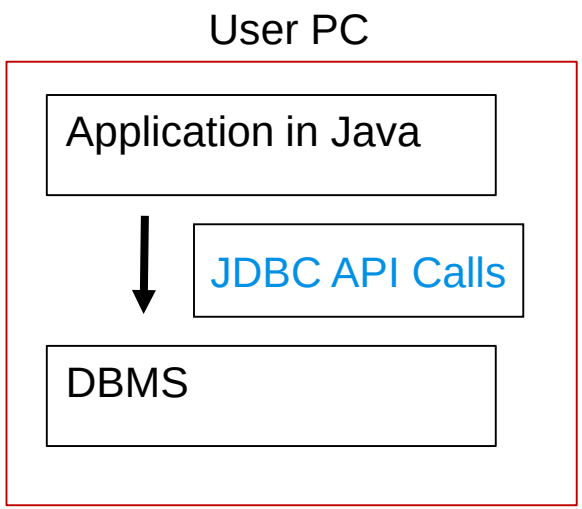
(a) Mainframe Era

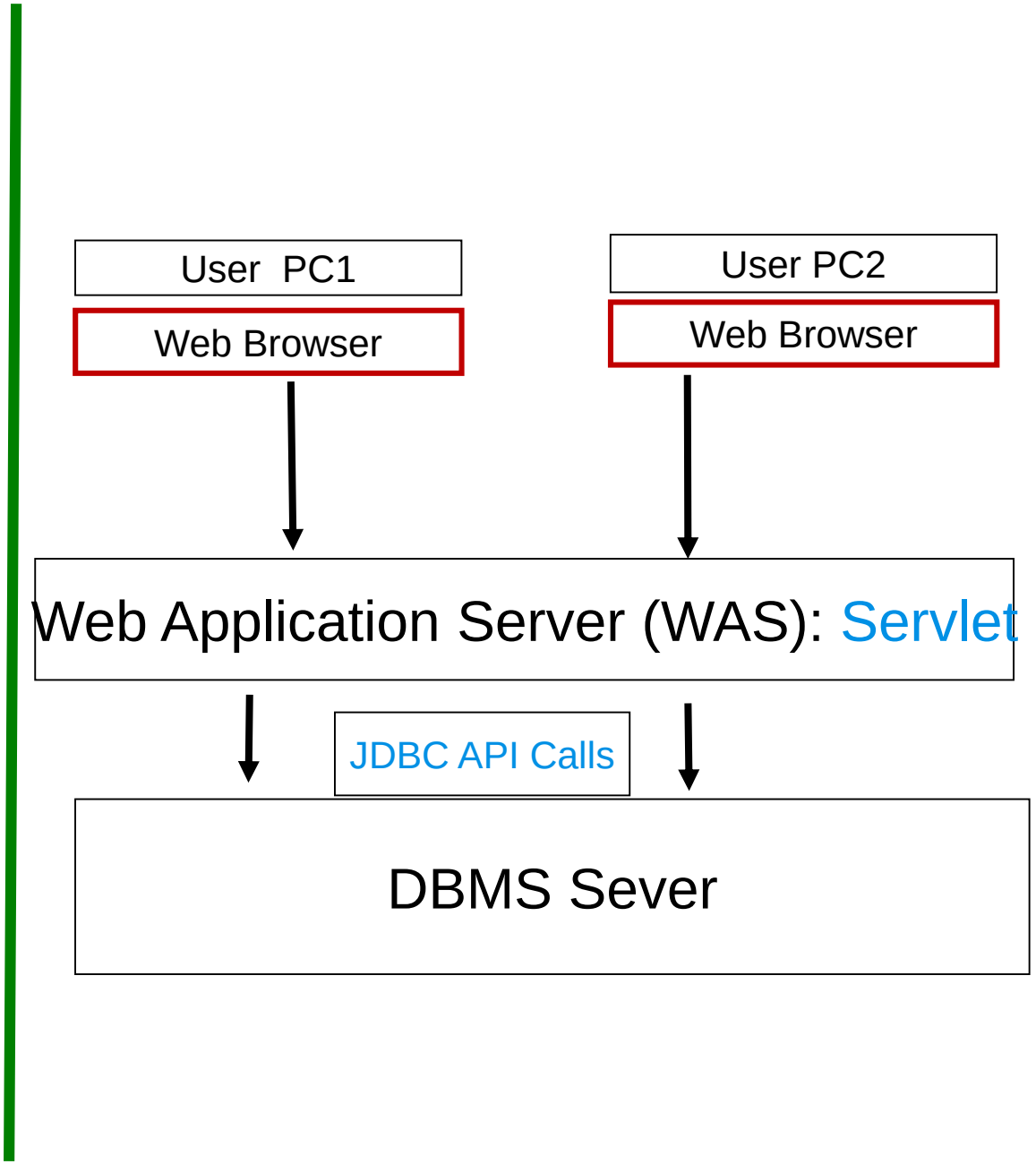


(b) Personal Computer Era



(c) Web era


User PC
Application in Java
JDBC API Calls
DBMS


User PC1
Web Browser
User PC2
Web Browser
Web Application Server (WAS): Servlet
JDBC API Calls
DBMS Sever

Application Programs and User Interfaces

- Most database users do *not* use a query language like SQL
- An application program acts as the intermediary between users and the database
  - Applications split into
    - front-end
    - middle layer
    - Backend

- Front-end: User interface
  - Forms
  - Graphical user interfaces
  - Many interfaces are Web-based
- Middle Layer: Application Server + Web Server or Web-Application Server (WAS)
- Backend: Database server

Web Interface

- Web browsers have become the de-facto standard user interface to databases
  - Enable large numbers of users to access databases from anywhere
  - Javascript, Flash and other scripting languages run in browser, but are downloaded transparently
  - Examples: banks, airline and rental car reservations, university course registration and grading, an so on

- The WWW is a distributed information system based on hypertext
- Most Web documents are hypertext documents formatted via the HyperText Markup Language (HTML)
- HTML documents contain
  - text along with font specifications, and other formatting instructions
  - hypertext links to other documents, which can be associated with regions of the text
  - forms, enabling users to enter data which can then be sent back to the Web server

Web Interfaces to Database

- Dynamic generation of HTML documents with data in database
  - Limitations of static HTML documents
    - Cannot customize fixed Web documents for individual users
    - Problematic to update Web documents, especially if multiple Web documents replicate data
  - Solution: Generate Web documents dynamically from data stored in a database
    - Can tailor the display based on user information stored in the database
      - E.g., tailored ads, tailored weather and local news, …
    - Displayed information is up-to-date, unlike the static Web pages
      - E.g., stock market information, ..
  - Therefore, web-application server generates HTML with query results and send the generated HTML to the browser

Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Uniform Resources Locators

- In the Web, pointing to a file is provided by Uniform Resource Locators (URLs)
- URL example:

  http://www.acm.org/sigmod

  - The first part indicates how the document is to be accessed (http: the Hyper Text Transfer Protocol)
  - The second part gives the unique name of a machine on the Internet
  - The rest of the URL identifies the document within the machine
- The local identification can be:
  - The path name of a file on the machine, or
  - An identifier (path name) of a program, plus arguments to be passed to the program
    - E.g., http://www.google.com/search?q=silberschatz
    - Google 창에 입력을 한 string을 Google URL에 variable q에 bind해서 보냄

HTML and HTTP

- HTML provides formatting, hypertext link, and image display features
  - including tables, stylesheets (to alter default formatting), etc.
- HTML is a collection of built-in tags
- HTML also provides tags for input features (accepting data and events from the user)
  - Select from a set of options
    - Pop-up menus, radio buttons, check lists
  - Enter values
    - Text boxes
  - Filled in input sent back to the server, to be acted upon by an executable at the server
- HyperText Transfer Protocol (HTTP) used for communication with the Web server

Sample HTML Source Text

ID	Name	Department
00128	Zhang	Comp. Sci.
12345	Shankar	Comp. Sci.
19991	Brandt	History

```
<html>
<body>
  <table border>
     <tr> <th>ID</th> <th>Name</th> <th>Department</th> </tr>
     <tr> <td>00128</td> <td>Zhang</td> <td>Comp. Sci.</td> </tr>
     ….
  </table>
  <form action="PersonQuery"   method=get>
     Search for:
        <select name="persontype">
           <option value="student" selected> Student  </option>
           <option value="instructor">  Instructor  </option>
        </select> <br>
     Name: <input type=text size=20 name="name">
     <input type=submit value="submit">
  </form>
</body>
</html>
```

Search for: Student
Name:
submit

Get 방식: 매개변수를 포함한 URL을 웹 서버에 요청
Post 방식: 매개변수만 HTTP 프로토콜을 통해 브라우저로 전송

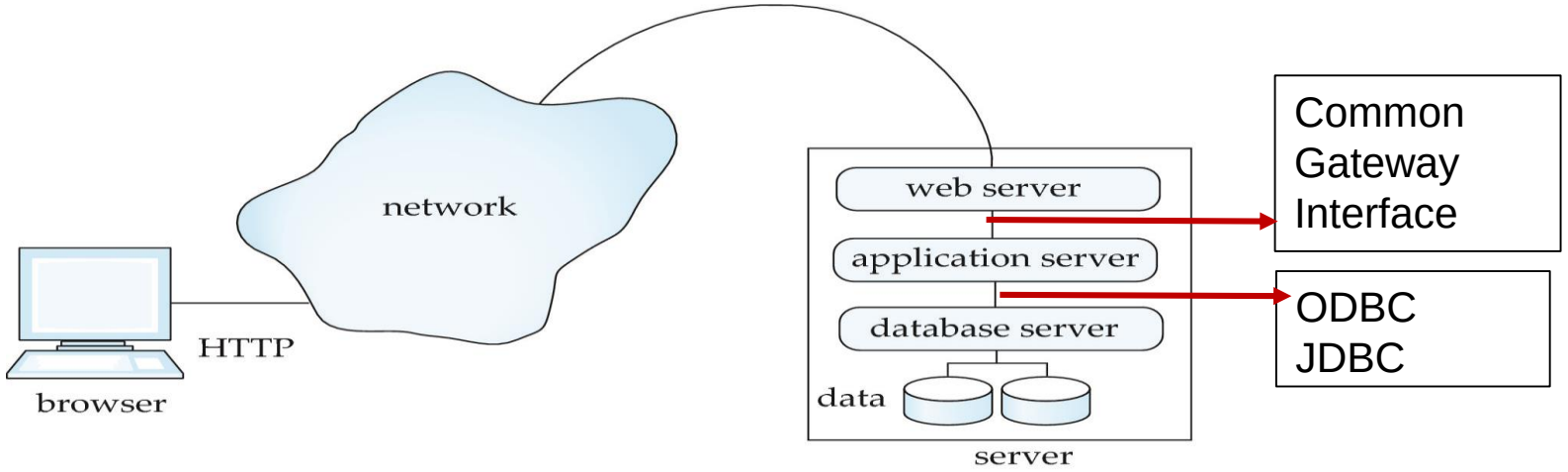
Web Servers

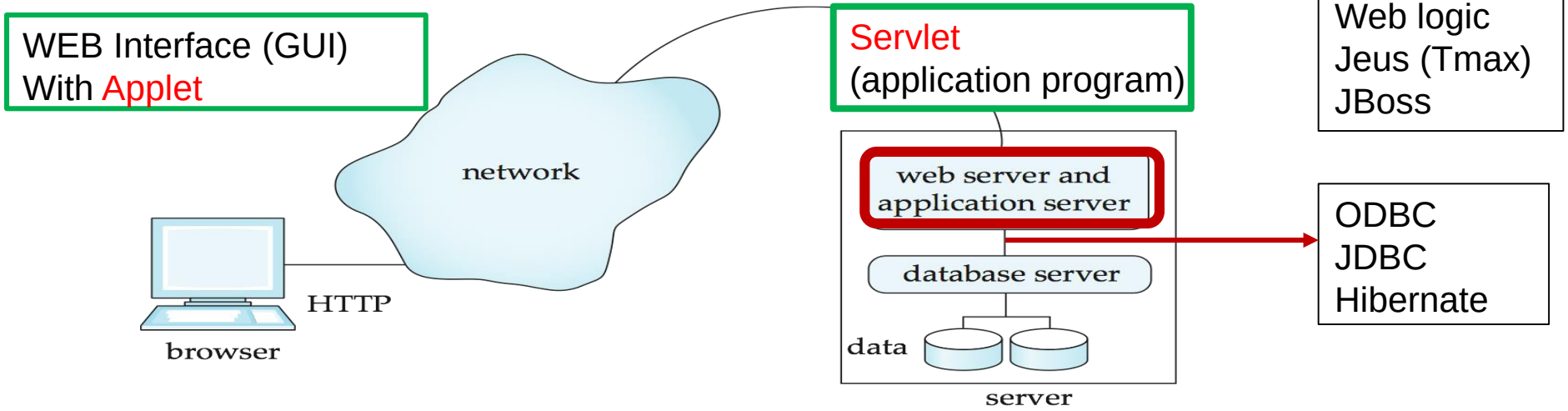
- A Web server can easily serve as a front end to a variety of information services
  - The document name in a URL may identify an executable program, that, when run, generates a HTML document
  - When an HTTP server receives a request for such a document, it executes the program, and sends back the HTML document that is generated
  - The Web client can pass extra arguments with the name of the document
- The Web browser provides a graphical user interface to the information service
- To install a new service on the Web, one simply needs to create and install an executable code that provides that service

- 3-layer Web Application Architecture
  - Common Gateway Interface (CGI): a standard interface between web server and application server
- 2-layer Web Application Architecture
  - Application program runs within the web server which we call "WAS" (Web Application Server)

2-Layer & 3-Layer Web Architecture

- Multiple levels of indirection have overheads





HTTP

- The HTTP protocol is **connectionless**
  - That is, once the server replies to a request, the server closes the connection with the client, and forgets all about the request
  - In contrast, Unix logins, and JDBC/ODBC connections stay connected until the client disconnects
    - retaining user authentication and other information
  - Motivation: reduces load on server
    - operating systems have tight limits on number of open connections on a machine
- Information services need session information
  - E.g., user authentication should be done only once per session
  - Solution: use a **cookie**

Sessions and Cookies in HTTP

- A **cookie** is a small piece of text containing identifying information
  - Sent by server to browser
    - Sent on first interaction, to identify session
  - Sent by browser to the server that created the cookie on further interactions
    - part of the HTTP protocol
  - Server saves information about cookies it issued, and can use it when serving a request
    - E.g., authentication information, and user preferences
- Cookies can be stored permanently or for a limited time

- 응용프로그램은 사용자 세션을 추적하기 위해 세션 식별자를 생성하고, 쿠키에 세션 식별자를 담아 서버로 보냄
- 서버는 클라이언트 요청이 왔을 때 클라이언트가 쿠키를 갖고 있지 않거나 서버에 저장된 것과 다른 값이 쿠키에 들어있을 경우 그 요청이 현재 세션에 속하지 않는 것으로 판단함

Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

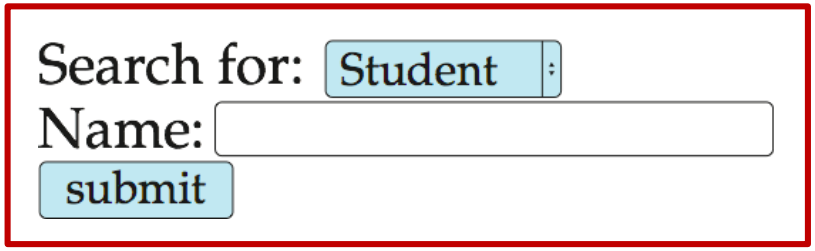
Servlets

- Java Servlet specification defines an API for communication between the Web-Application Server and application program running in the server
  - E.g., methods to get parameter values from Web forms, and to send HTML text back to client
- Application program (also called a servlet) is loaded into the server
  - Each request spawns a new thread in the server
  - The thread is closed once the request is serviced
- Servlet API provides a getSession() method
  - Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
  - Provides methods to store and look-up per-session information
    - E.g. user name, preferences, ..
- Servlet API calls and JDBC API calls are mixed in a servlet program

HTML code with a form tag with PersonQuery in the Client Side Brower

```
<form action="PersonQuery" method=get>
    Search for:
      <select name = "persontype" >
        <option value = "student" selected> Student </option>
        <option value = "instructor"> Instructor </option>
      </select> <br>
    Name: <input type=text size=20 name = "name" >
    <input type=submit value="submit">
 </form>
</body>
</html>
```

어떤 URL을 open해서 아래 창이 있다면



** action attribute of form tag: form이 채워지면 form에 있는 data를 URL PersonQuery로 보냄
(persontype과 name에 사용자가 입력한 data를 실어서)
** PersonQuery.class 혹은 PersonQuery.jsp 가 있는 server로 실행을 request

- HTML로만 구성된 Client-side application program
- HTML 안에 Java Applet이나 JavaScript등으로 dynamic operation이 가능하다
- Server-side에 있는 PersonQuery servlet을 match는 Web Application Server (WAS)의 몫

PersonQueryServlet Code [1/2] in the Server Side (Java Servlet)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class PersonQueryServlet extends HttpServlet {
   public void doGet (HttpServletRequest request, HttpServletResponse response)
                                 throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<HEAD><TITLE> Query Result </TITLE></HEAD>");
      out.println("<BODY>");
        ….. BODY OF SERVLET (next slide) …
      out.println("</BODY>");
      out.close();
   }
}
```

- 앞페이지에 있는 PersonQuery form이 WAS에 도착하면 해당 servlet인 PersonQueryServlet.class를 구동시킨다 (JSP로 되어 있으면 PersonQueryServlet.jsp)
- Servlet 수행이 끝나면 out strea에 있는 HTML을 client로 보낸다

PersonQueryServlet Code [2/2]

```
String persontype = request.getParameter("persontype");
String number = request.getParameter("name");
if(persontype.equals("student")) {
    ... code to find students with the specified name ...
    ... using JDBC to communicate with the database ..
    out.println( "<table BORDER COLS=3>“ );
    out.println( " <tr> <td>ID</td> <td>Name: </td>" + " <td>Department</td> </tr>“ );
    for(... each result ...){
        ... retrieve ID, name and dept name
        ... into variables ID, name and deptname
        out.println( "<tr> <td>" + ID + "</td>" + "<td>" + name + "</td>" + "<td>" + deptname
                + "</td></tr>“ );
    };
    out.println( "</table>“ );
}
else {
    ... as above, but for instructors ...
}
```

WAS는 Servlet을 실행하면서 JDBC call을 DBMS에 pass하고
DBMS에서 return된 data를 가지고 HTML 를 생성하고
Servlet 실행완료후에는 생성된 HTML을 Client-side Browser로 pass한다

Servlet Sessions

- Servlet API supports handling of sessions
  - Sets a cookie on first interaction with browser, and uses it to identify session on further interactions
- To check if session is already active:
  - if (request.getSession(false) == true)
    - then existing session
    - else redirect to authentication page
  - authentication page
    - check login/password
    - request.getSession(true): creates new session
- Store/retrieve attribute value pairs for a particular session
  - session.setAttribute(“userid”, userid)
  - session.getAttribute(“userid”)

Servlet Support in WAS

- Servlets run inside application servers (WAS) such as
  - Apache Tomcat, Glassfish, Jboss, Jeus (Tmax)
  - BEA Weblogic, IBM WebSphere and Oracle Application Servers

- Modern WAS supports Java 2 Enterprise Edition (J2EE) Platform
  - 분산객체 환경에서 서버측 App개발 플랫폼
    - Servlet/JSP
    - JDBC
    - JMS (Java Messaging)
    - JNDI (Naming & Directory
    - JTA (Java Transaction)
    - TCP/IP, HTTP, Etc (Protocol)
    - Java RMI (Distributed Object Framework)

Java SE8 (JDK8): Java Platform, Standard Edition

Server-Side Scripting

- 간단한 웹 응용 프로그램을 작성한다고 해도 이를 Java나 C를 연동해 작성하는 일은 상대적으로 어려움
- Server-side scripting simplifies the task of connecting a database to the Web
  - Define an HTML document with embedded executable code/SQL queries
  - Input values from HTML forms can be used directly in the embedded code/SQL queries
  - When the document is requested, the Web server executes the embedded code/SQL queries to generate the actual HTML document
- Numerous server-side scripting languages
  - Web-programming purpose script language
    - JSP (JavaServer Page, 1999)
    - ASP (ActiveServer Page, 2000)
    - PHP (*PHP: Hypertext Preprocessor, 1995)*
    - ColdFusion's Markup Language (cfml), Jscript, and so on
  - General purpose scripting languages
    - Python, VBScript, Perl, Python

JSP (JavaServer Pages)

- A JSP page with embedded Java code (very simple for programmer)

hello.jsp

```
<html>
<head> <title> Hello </title> </head>
<body>
<% if (request.getParameter("name") == null)
                { out.println("Hello World"); }
    else        { out.println("Hello, " + request.getParameter("name")); }
%>
</body>
</html>
```

- When a JSP page is requested by a bowser
  - WAS first executes the Java code inside the red box
  - And generates HTML output from the JSP page
  - The generated HTML output is sent back to the browser

- JSP allows new tags to be defined in tag libraries
  - to build rich user interfaces such as paginated display of large datasets

PHP (PHP: Hypertext Preprocessor)

- PHP is widely used for Web server scripting
- Extensive libaries including for database access using ODBC

hello.php

```
<html>
   <head> <title> Hello </title> </head>
   <body>
   <?php   if (!isset($_REQUEST[‘name’]))
                  { echo “Hello World”; }
            else { echo “Hello, ” + $_REQUEST[‘name’]; }
   ?>
   </body>
</html>
```

Client Side Scripting

- Executing programs at the client site speeds up interaction by avoiding many round trips to server
  - 기본적으로 서버의 부담을 줄이기 위해 클라이언트 측에서 수행하는 스크립트
- Client-side scripts/programs allow documents to be active and dynamic
  - E.g., animation by executing programs at the local site
  - E.g., ensure that values entered by users satisfy some correctness checks
  - Permit flexible interaction with the user

- Browsers can fetch certain scripts (client-side scripts) or programs along with documents, and execute them in "safe mode" at the client site
  - Javascript
  - Applets (using Java)
  - Macromedia Flash and Shockwave (for animation/games)
  - VRML

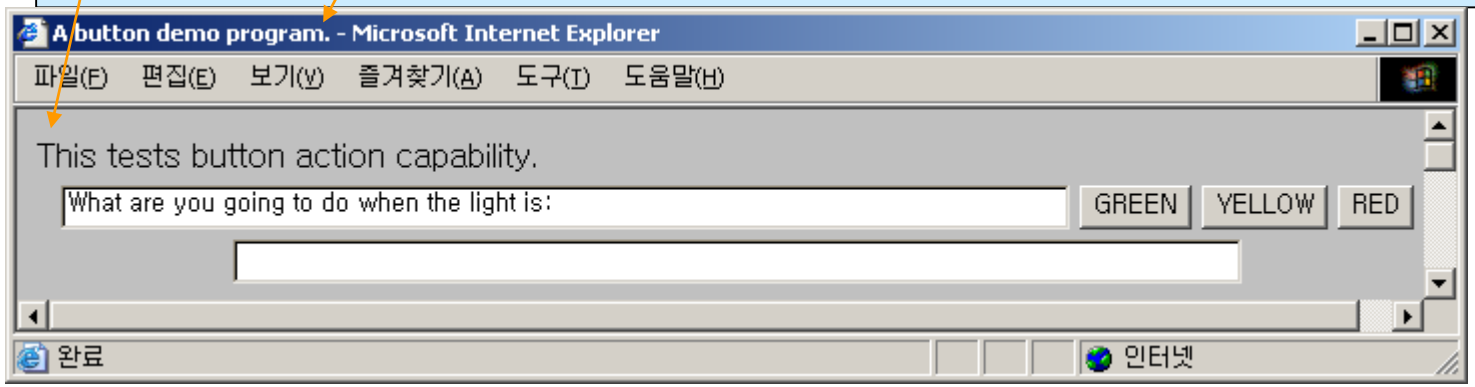
** 혹시 Applet이 Servlet과 1대1 대응된다고 생각하면 안됨

Java Applet inside HTML [1/3]

- Show Green, Red, Yello button on the screen
- If the user clicks any button, trigger the action method by showing the corresponding message (“stop”, “go”, “ready to stop”)

TrafficLight.htm

```
<HTML>
<HEAD>  <TITLE> A button demo program. </TITLE>  </HEAD>
    <BODY>
    This tests button action capability.
    <APPLET code="TrafficLight.class" WIDTH=700 HEIGHT=325>  </APPLET>
    </BODY>
</HTML>
```



Java Applet inside HTML [2/3]

```
import java.awt.*;
import java.awt.event.*;

public class TrafficLight extends java.applet.Applet
                          implements ActionListener
{
   TextField   m1, m2;
   Button      b1, b2, b3;

   public void init ()
   {
      m1 = new TextField(80);
      m1.setText("What are you going to do when the light is:");
      b1 = new Button("GREEN");
      b2 = new Button("YELLOW");
      b3 = new Button("RED");
      m2 = new TextField(80);
      add(m1) ;
      add(b1) ;
      add(b2) ;
      add(b3) ;
      add(m2) ;
      b1.addActionListener(this);
      b2.addActionListener(this);
      b3.addActionListener(this);
   }
```

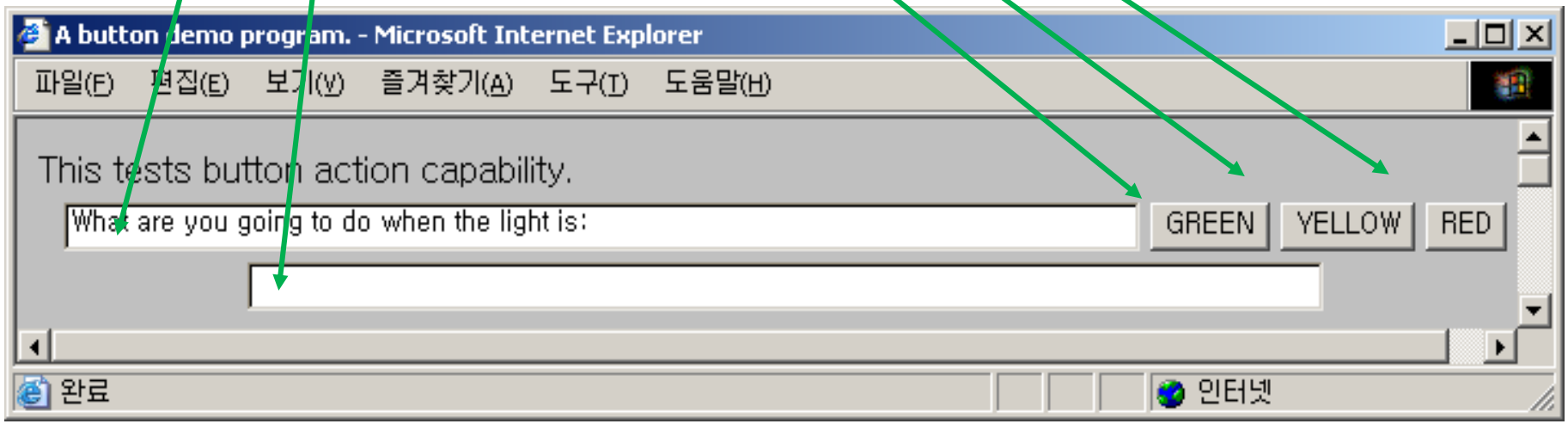
```
   public void actionPerformed(ActionEvent event)
   {
      Object cause = event.getSource();

      if (cause == b1)
      {
         m2.setText("Keep on rolling.");
      }
      if (cause == b2)
      {
         m2.setText("Step on it!  You can make it!");
      }
      if (cause == b3)
      {
         m2.setText("I suppose you'll have to stop.");
      }
   }
}
```

TrafficLight.java

Java Applet inside HTML [3/3]

```
public class TrafficLight extends java.applet.Applet
                          implements ActionListener  {
  TextField   m1, m2;
  Button      b1, b2, b3;
  public void init ()
  {   m1 = new TextField(80);
      m1.setText("What are you going to do when the light is:");
      b1 = new Button("GREEN");
      b2 = new Button("YELLOW");
      b3 = new Button("RED");
      m2 = new TextField(80);
      add(m1) ;
      add(b1) ;
      add(b2) ;
      add(b3) ;
      add(m2) ;
```



Client Side Scripting and Security

- Applet을 외부에서 다운받아서 browser에서 실행
  - Malicious Appliet can access local data of your PC

- Security mechanisms needed to ensure that malicious scripts do not cause damage to the client machine
  - Easy for limited capability scripting languages
  - Harder for general purpose programming languages like Java

- E.g., Java's security system (such as SandBox) ensures that the Java applet code does not make any system calls directly
  - Disallows dangerous actions such as file writes
  - Notifies the user about potentially dangerous actions, and allows the option to abort the program or to continue execution.

Javascript

- Javascript very widely used (Web App with Frequent User Interactions에 적합)
  - forms basis of new generation of Web applications (called Web 2.0 applications) offering rich user interfaces
- Javascript functions for Web App with Frequent User Interactions
  - check **input for validity**
  - modify the displayed Web page, by altering the underling **document object model (DOM)** tree representation of the displayed HTML text
  - communicate with a Web server **to fetch data** and modify the current page using **fetched data**, without needing to reload/refresh the page
    - forms basis of AJAX technology used widely in Web 2.0 applications
    - E.g. on selecting a country in a drop-down menu, the list of states in that country is automatically populated in a linked drop-down menu

◆ AJAX(Asynchronous Javascript and XML)
기존의 WEB APP이 Client 요청에 대해 데이터를 가공하여 HTML Web Page 를 return, Ajax는 SOAP 프로토콜을 사용하여 데이터만을 client에게 XML형식으로 return

◆Web 2.0 : 사용자들의 소통과 정보공유를 제공하는 Web Application Platform
사례) UCC, Service Wikis, Social Network Individual

Input Validation Checking in Javascript

- Example of Javascript used to validate form input

```
<html>
<head>
    <script type = "text/javascript">
        function validate() {
            var credits = document.getElementById("credits").value;
            if ( isNaN(credits) || credits<=0 || credits>=16) {
                alert("Credits must be a number greater than 0 and less than 16");
                return false
            }
        }
    < /script>
</head>
<body>
    <form action= "createCourse"  onsubmit="return validate()">
        Title: <input type="text" id="title" size="20"><br />
        Credits: <input type="text" id="credits" size="2"><br />
        <Input type="submit" value="Submit">
    < /form>
</body>
</html>
```

Title:

Credit:

Submit

Ajax Application in JavaScript

참고자료

- JQuery : Javascript library
  - Ajax 개발, 코드 보기, DOM 찾기, 애니메이션 만들기, 이벤트 제어 등을 위한 기능 제공

Client program with AJAX

demo1.html

```
1  <p>time : <span id="time"></span></p>
2  <input type="button" id="execute" value="execute" />
3  <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>
4  <script>
5      $('#execute').click(function(){
6          $.ajax({
7              url:'./time.php',
8              success:function(data){
9                  $('#time').append(data);
10             }
11         })
12     })
13 </script>
```

Only Data

성공시 호출할 콜백

Servlet

time.php

```
1  <?php
2  $d1 = new DateTime;
3  $d1->setTimezone(new DateTimezone("asia/seoul"));
4  echo $d1->format('H:i:s');
5  ?>
```

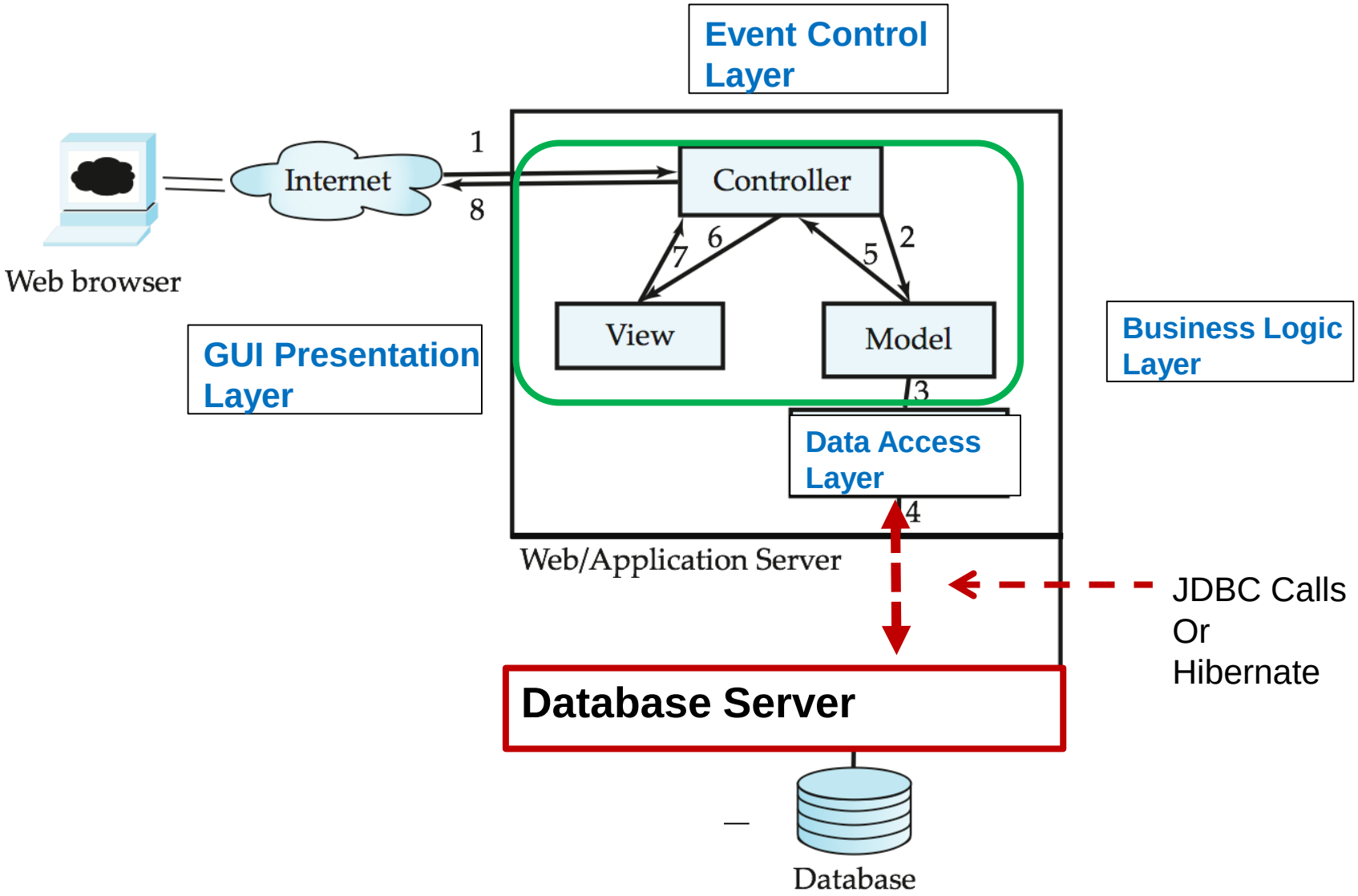
Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Application Architectures

- Application layers
  - **Presentation or user interface** layer
    - **model-view-controller (MVC) architecture**
      - **model**: business logic
      - **view**: presentation of data, depends on display device
      - **controller**: receives events, executes actions, and returns a view to the user
  - **business-logic** layer
    - provides high level view of data and actions on data
      - often using an object data model
    - hides details of data storage schema
  - **data access** layer
    - interfaces between business logic layer and the underlying database
    - provides mapping from object model of business layer to relational model of database

Fig 9.11: Application Architecture with MVC Paradigm



Business Logic Layer

- Provides abstractions of entities
  - e.g. students, instructors, courses, etc
- Enforces **business rules** for carrying out actions
  - E.g. student can enroll in a class only if she has completed prerequsites, and has paid her tuition fees
- Supports **workflows** which define how a task involving multiple participants is to be carried out
  - E.g. how to process application by a student applying to a university
  - Sequence of steps to carry out task
  - Error handling
    - e.g. what to do if recommendation letters not received on time
  - Workflows discussed in Section 26.2

The Data-Access Layer
(Object-Relational Mapping)

- Most application code written in OOPL (Java, Python), while storing data in a traditional relational database
- Schema designer has to provide a mapping between object data and relational schema
  - e.g. Java class *Student* mapped to relation *student,* with corresponding mapping of attributes
  - An object can map to multiple tuples in multiple relations
- Application opens a session, which connects to the database
- Objects can be created and saved to the database using session.save(object)
  - mapping used to create appropriate tuples in the database
- Query can be run to retrieve objects satisfying specified predicates

- Object-Relational Mapping이란 요청이 있을 때마다 생성된 서버 메모리상의 객체와 DB 릴레이션의 데이터를 자동적으로 mapping시켜 주는 방식

Java Application with JDBC Code in Chapter 5

참고자료

```
public static void JDBCexample(String userid, String passwd)
{ try {
        Class.forName ("oracle.jdbc.driver.OracleDriver"); // JDBC driver loading
        Connection conn = DriverManager.getConnection(
                        "jdbc:oracle:thin:@db.yale.edu:2000:univdb", userid, passwd);
      Statement stmt = conn.createStatement();
      ResultSet rset = stmt.executeQuery(
           "select dept_name, avg (salary)
            from instructor
            group by dept_name");
      while (rset.next()) {
           System.out.println( rset.getString("dept_name") + " " + rset.getFloat(2) );
       }
        stmt.close();
        conn.close();
 }
 catch (Exception e) {    System.out.println("Exception : " + e);  }
}
```

Object-Relational Mapping Systems

- The **Hibernate ORM** object-relational mapping system is widely used (open source)
  - public domain system, runs on a variety of database systems
  - supports a query language that can express complex queries involving joins
    - translates queries into SQL queries
  - allows relationships to be mapped to sets associated with objects
    - e.g. courses taken by a student can be a set in Student object
  - See book for Hibernate code example

- Hibernate 시스템은 Java 객체와 DB의 릴레이션을 맵핑시켜 줌
- Java 클래스와 릴레이션의 연결은 mapping file에 명시되어 있음

- The **Entity Data Model** developed by Microsoft
  - provides an entity-relationship model directly to application
  - maps data between entity data model and underlying storage, which can be relational
  - Entity SQL language operates directly on Entity Data Model

HYBERNATE EXAMPLE: Create Java class from Student relation [1/2]

```
public class Student {
        String ID;
        String name;
        String department;
        int tot_cred;
        Student(String id, String name, String dept, int totcreds); // constructor
}
```

To be precise, the class attributes should be declared as private, and getter/setter methods should be provided to access the attributes, but we omit these details.

The mapping of the class attributes of Student to attributes of the relation *student* is specified in a mapping file, in an XML format. Again, we omit details.

student 릴레이션과 대응하는 Java 클래스를 생성

자바의 Student 클래스와 DB의 Student 릴레이션의 속성정보에 대한 연결은 mapping file에 XML형식으로 명시

The following code snippet then creates a Student object and saves it to the database.

```
Session session = getSessionFactory().openSession();
Transaction txn = session.beginTransaction();
Student stud = new Student("12328", "John Smith", "Comp. Sci.", 0);
session.save(stud);
txn.commit();
session.close();
```

Student 객체를 생성하고 save() 함수를 통해서 데이터베이스에 튜플로 저장하고 있음

이 때 필요한 SQL insert문은 자동적으로 생성되어 수행됨

HYBERNATE EXAMPLE: Create Java class from Student relation [2/2]

```
public class Student {
        String ID;
        String name;
        String department;
        int tot_cred;
        Student(String id, String name, String dept, int totcreds); // constructor
}
```

Hibernate automatically generates the required SQL **insert** statement to create a *student* tuple in the database.

To retrieve students, we can use the following code snippet

```
Session session = getSessionFactory().openSession();
Transaction txn = session.beginTransaction();
List students =
        session.find("from Student as s order by s.ID asc");
for ( Iterator iter = students.iterator(); iter.hasNext(); ) {
        Student stud = (Student) iter.next();
        .. print out the Student information ..
}
txn.commit();
session.close();
```

The above code snippet uses a query in Hibernate's HQL query language. The HQL query is automatically translated to SQL by Hibernate and executed, and the results are converted into a list of Student objects. The for loop iterates over the objects in this list and prints them out.

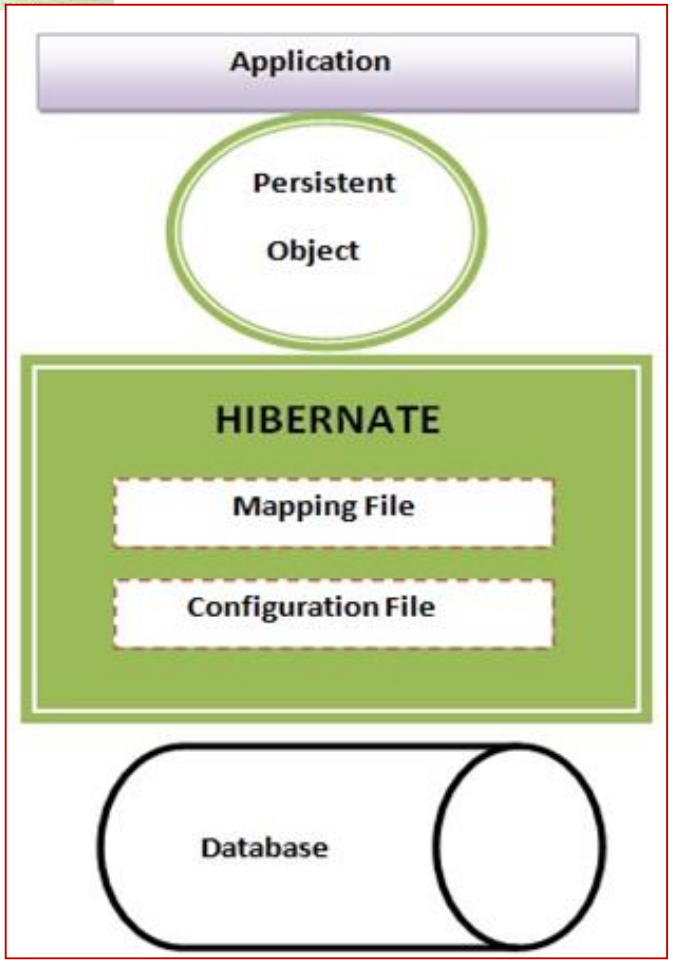
아래 코드는 iterator를 사용해 Student를 추출하고 있는 HQL 질의어임

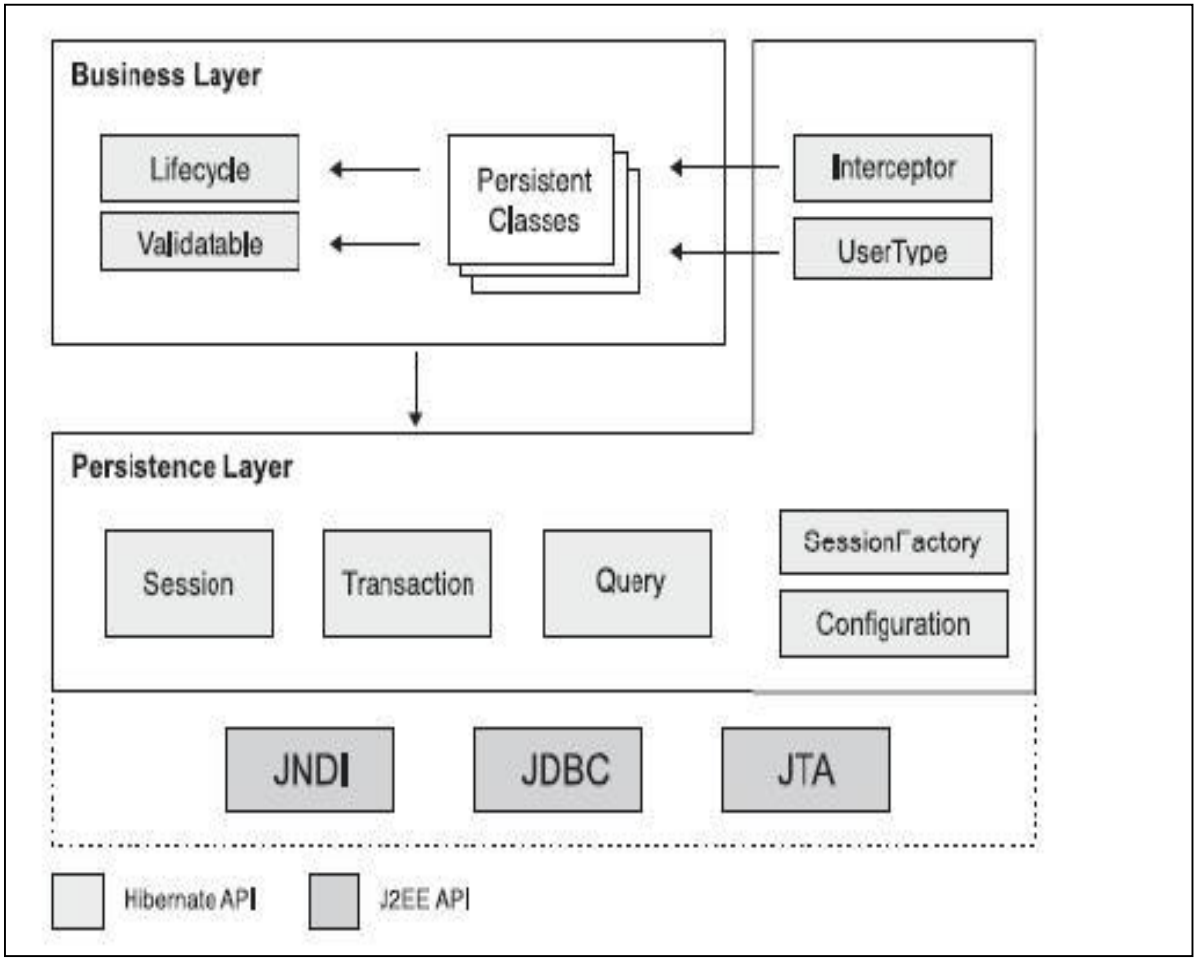
자바 스타일의 HQL은 Hibernate내에서 자동적으로 SQL로 변환되어 수행되고, 그 결과 또한 자바의 class로 반환됨

즉 DB의 Student 릴레이션의 각 튜플들을 자바의 Student class로 추출하고 있는 코드임

참고자료

Hibernate Architecture





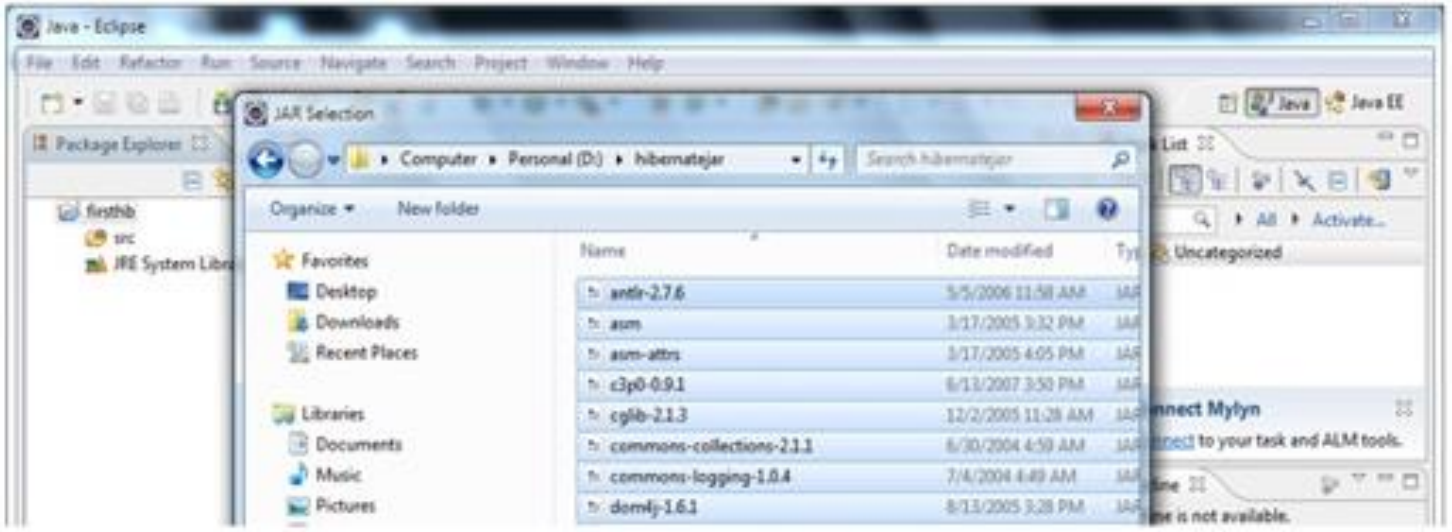
- Hibernate 시스템 내에서 session을 호출함으로써 DB의 instance들을 persistent object화 시킴. // Persistent object란 간단히 말해 자바 상의 객체, 즉 사용자가 정의한 클래스를 뜻함
- Persistent object는 session 내에서 transactional하게 수행될 수 있고, (어플리케이션에서 수행한) 트랜잭션이 종료된 후 다시 DB에 싱크를 맞춰서 저장하게 됨
- Persistent Object는 다수의 클라이언트 환경을 지원하기 위해 자동적인 dirty checking, 변경된 데이터에 대한 dynamic updates 등이 지원됨

Hibernate Examples [1/6]

참고자료

1. Create Java Project and Add jar files for Hibernate
2. Create Persistent Class and Database Table
3. Create Mapping file
4. Create Configuration file
5. Create Application Class
6. Run the Application

- Create Java Project and Add jar files for Hibernate



Hibernate Examples [2/6]

참고자료

- Create Persistent Class

```
public class Employee {
   private int id;
   private String firstName;
   private String lastName;
   private int salary;

   public Employee() {}
   public Employee(String fname, String lname, int salary) {
      this.firstName = fname;
      this.lastName = lname;
      this.salary = salary;
   }
   public int getId() {
      return id;
   }
   public void setId( int id ) {
      this.id = id;
   }
   public String getFirstName() {
      return firstName;
   }
   public void setFirstName( String first_name ) {
      this.firstName = first_name;
   }
   public String getLastName() {
      return lastName;
   }
   public void setLastName( String last_name ) {
      this.lastName = last_name;
   }
   public int getSalary() {
      return salary;
   }
   public void setSalary( int salary ) {
      this.salary = salary;
   }
}
```

Key 역할에 대응하는 속성(필수)

자바프로젝트 내에서
자바 소스 파일로 생성

Employee.java

- Create Database Tables

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   salary     INT  default NULL,
   PRIMARY KEY (id)
);
```

자바의 Employee 객체와 대응할
DB내의 Employee 테이블을 생성

Hibernate Examples [3/6]

참고자료

- Create Mapping file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <meta attribute="class-description">
         This class contains the employee detail.
      </meta>
      <id name="id" type="int" column="id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="salary" column="salary" type="int"/>
   </class>
</hibernate-mapping>
```

자바 객체와 DB 테이블 mapping

자바 객체

테이블 이름

(옵션) 클래스 설명

자바 객체의 키 역할의 속성과 DB Table의 primary key를 mapping

자바 객체의 속성

테이블의 column 이름

Key 아닌 나머지 속성들 mapping

Employee.hbm.xml

Hibernate Examples [4/6]

참고자료

- Create Configuration file

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
   <property name="hibernate.dialect">
      org.hibernate.dialect.MySQLDialect
   </property>
   <property name="hibernate.connection.driver_class">
      com.mysql.jdbc.Driver
   </property>

   <!-- Assume test is the database name -->
   <property name="hibernate.connection.url">
      jdbc:mysql://localhost/test
   </property>
   <property name="hibernate.connection.username">
      root
   </property>
   <property name="hibernate.connection.password">
      root123
   </property>

   <!-- List of XML mapping files -->
   <mapping resource="Employee.hbm.xml"/>

</session-factory>
</hibernate-configuration>
```

앞서 작성한 mapping file을 명시함(Employee.hbm.xml)

Hibernate.cfg.xml

Hibernate Examples [5/6]

참고자료

- Create Application class (1)

ManageEmployee.java

```
import java.util.List;
import java.util.Date;
import java.util.Iterator;

import org.hibernate.HibernateException;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class ManageEmployee {
   private static SessionFactory factory;
   public static void main(String[] args) {
      try{
         factory = new Configuration().configure().buildSessionFactory();
      }catch (Throwable ex) {
         System.err.println("Failed to create sessionFactory object." + ex);
         throw new ExceptionInInitializerError(ex);
      }
      ManageEmployee ME = new ManageEmployee();

      /* Add few employee records in database */
      Integer empID1 = ME.addEmployee("Zara", "Ali", 1000);
      Integer empID2 = ME.addEmployee("Daisy", "Das", 5000);
      Integer empID3 = ME.addEmployee("John", "Paul", 10000);

      /* List down all the employees */
      ME.listEmployees();

      /* Update employee's records */
      ME.updateEmployee(empID1, 5000);

      /* Delete an employee from the database */
      ME.deleteEmployee(empID2);

      /* List down new list of the employees */
      ME.listEmployees();
   }
```

- Run the Application

- Employee.java
- Employee.hbm.xml
- Hibernate.cfg.xml
- ManageEmployee.java

위 파일들이 모두 적절한 경로에 존재해야 함

실행결과

```
First Name: Zara  Last Name: Ali  Salary: 1000
First Name: Daisy  Last Name: Das  Salary: 5000
First Name: John  Last Name: Paul  Salary: 10000
```

```
First Name: Zara  Last Name: Ali  Salary: 5000
First Name: John  Last Name: Paul  Salary: 10000
```

Hibernate Examples [6/6]

참고자료

- Create Application class (2)

ManageEmployee.java

```
/* Method to CREATE an employee in the database */
public Integer addEmployee(String fname, String lname, int salary){
   Session session = factory.openSession();
   Transaction tx = null;
   Integer employeeID = null;
   try{
      tx = session.beginTransaction();
      Employee employee = new Employee(fname, lname, salary);
      employeeID = (Integer) session.save(employee);
      tx.commit();
   }catch (HibernateException e) {
      if (tx!=null) tx.rollback();
      e.printStackTrace();
   }finally {
      session.close();
   }
   return employeeID;
}
```

```
/* Method to UPDATE salary for an employee */
public void updateEmployee(Integer EmployeeID, int salary ){
   Session session = factory.openSession();
   Transaction tx = null;
   try{
      tx = session.beginTransaction();
      Employee employee =
                (Employee)session.get(Employee.class, EmployeeID);
      employee.setSalary( salary );
              session.update(employee);
      tx.commit();
   }catch (HibernateException e) {
      if (tx!=null) tx.rollback();
      e.printStackTrace();
   }finally {
      session.close();
   }
}
```

```
/* Method to  READ all the employees */
public void listEmployees( ){
   Session session = factory.openSession();
   Transaction tx = null;
   try{
      tx = session.beginTransaction();
      List employees = session.createQuery("FROM Employee").list();
      for (Iterator iterator =
                        employees.iterator(); iterator.hasNext();){
         Employee employee = (Employee) iterator.next();
         System.out.print("First Name: " + employee.getFirstName());
         System.out.print("  Last Name: " + employee.getLastName());
         System.out.println("  Salary: " + employee.getSalary());
      }
      tx.commit();
   }catch (HibernateException e) {
      if (tx!=null) tx.rollback();
      e.printStackTrace();
   }finally {
      session.close();
   }
}
```

```
   /* Method to DELETE an employee from the records */
   public void deleteEmployee(Integer EmployeeID){
      Session session = factory.openSession();
      Transaction tx = null;
      try{
         tx = session.beginTransaction();
         Employee employee =
                   (Employee)session.get(Employee.class, EmployeeID);
         session.delete(employee);
         tx.commit();
      }catch (HibernateException e) {
         if (tx!=null) tx.rollback();
         e.printStackTrace();
      }finally {
         session.close();
      }
   }
}
```

JDBC vs HIBERNATE [1/4]

참고자료

1) Load the RDBMS specific JDBC driver because this driver actually communicates with the DB
2) Open the connection to DB which is then used to send SQL statements and get results back
3) Create JDBC Statement object. This object contains SQL query.
4) Execute statement which returns resultset(s)
   - ResultSet contains the tuples of DB table as a result of SQL query
5) Process the result set
6) Close the connection

JDBC Code

Example: Retrieve list of employees from Employee table using JDBC.

```
String url = "jdbc:odbc:" + dbName;
List<EmployeeBean> employeeList = new ArrayList<EmployeeBean>();

/* load the jdbc-odbc driver */
class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

/* Open a connection to database */
Connection con = DriverManager.getConnection(url);

/* create Statement object */
Statement stmt = con.createStatement();

/* execute statement */
ResultSet rs = stmt.executeQuery("SELECT * FROM Sells");
while ( rs.next() )
{
      EmployeeBean eb = new Employeebean();
      eb.setName(rs.getString("name"));
      eb.setSalary(rs.getFloat("salary"));
      employeeList.add(eb);
}
```

ResultSet

rs ➡

id	name	salary
1	Tom	2000
2	Jack	1500
3	Mary	2500

JDBC vs HIBERNATE [2/4]

참고자료

1. Load the Hibernate configuration file and create configuration object.
   - It will automatically load all hbm mapping files.
2. Create session factory from configuration object
3. Get one session from this session factory
4. Create HQL query
5. Execute query to get list containing Java objects

HIBERNATE Code

Example: Retrieve list of employees from Employee table using Hibernate.

```
/* Load the hibernate configuration file */
Configuration cfg = new Configuration();
cfg.configure(CONFIG_FILE_LOCATION);

/* Create the session factory */
SessionFactory sessionFactory = cfg.buildSessionFactory();

/* Retrieve the session */
Session session = sessionFactory.openSession();

/* create query */
Query query = session.createQuery("from  EmployeeBean");

/* execute query and get result in form of Java objects */
List<EmployeeBean> finalList = query.list();
```

JDBC vs HYBERNATE [3/4]

참고자료

Object-Relation Mapping File

EmployeeBean.hbm.xml File

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
      <class name="com.mf.bean.EmployeeBean"
            table="t_employee">

            <id name="id" type="string" unsaved-value="null">
                  <column name="id" sql-type="varchar(32)" not-null="true"/>
                  <generator class="uuid"/>
            </id>

            <property name="name">
                  <column name="name" />
            </property>

            <property name="salary">
                  <column name="salary" />
            </property>
      </class>
</hibernate-mapping>
```

JDBC vs HYBERNATE [4/4]

참고자료

1) With JDBC, developer has to write code to map an object model's data representation to a relational data model and its corresponding database schema. Hibernate is flexible and powerful ORM solution to map Java classes to database tables.

2) Hibernate provides transparent persistence and developer does not need to write code explicitly to map database tables tuples to application objects during interaction with RDBMS. With JDBC this conversion is to be taken care of by the developer manually with lines of code.

3) Hibernate provides a powerful query language Hibernate Query Language (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for polymorphic queries. Hibernate also supports native SQL statements

4) As table changed or database changed then it’s essential to change object structure as well as to change code written to map table-to-object/object-to-table. Hibernate provides this mapping itself.

5) Hibernate is an open source and free to use for both development and production deployments.

What is Web Service? [1/2]

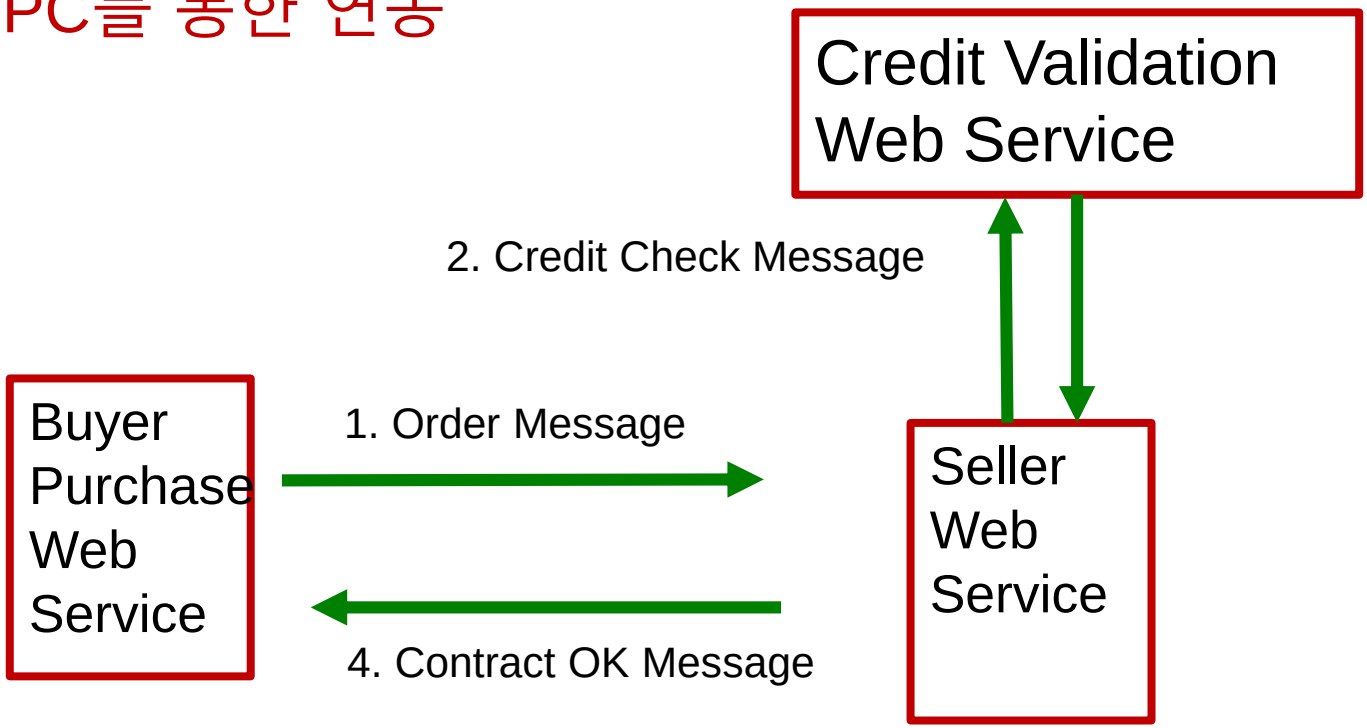
- *Web services* are client and server applications that communicate over the World Wide Web's (WWW) using HyperText Transfer Protocol (HTTP)
- As described by the World Wide Web Consortium (W3C), web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks
- Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML
- Web services can be combined in a loosely coupled way to achieve complex operations
- Programs providing simple services can interact with each other to deliver sophisticated added-value services

- Web Service란 서로 다른 컴퓨팅 환경에서 사용되는 모든 APP SW들이 직접 소통하고 실행될 수 있도록 Dynamic System 환경을 구현해 주는 SW

What is Web Service? [2/2]

- "Programmable application logic accessible using Standard Internet Protocols…" – Microsoft
- "An interface that describes a collection of operations that are network accessible through standardized XML messaging …" – IBM

SW와 SW간의
RPC를 통한 연동



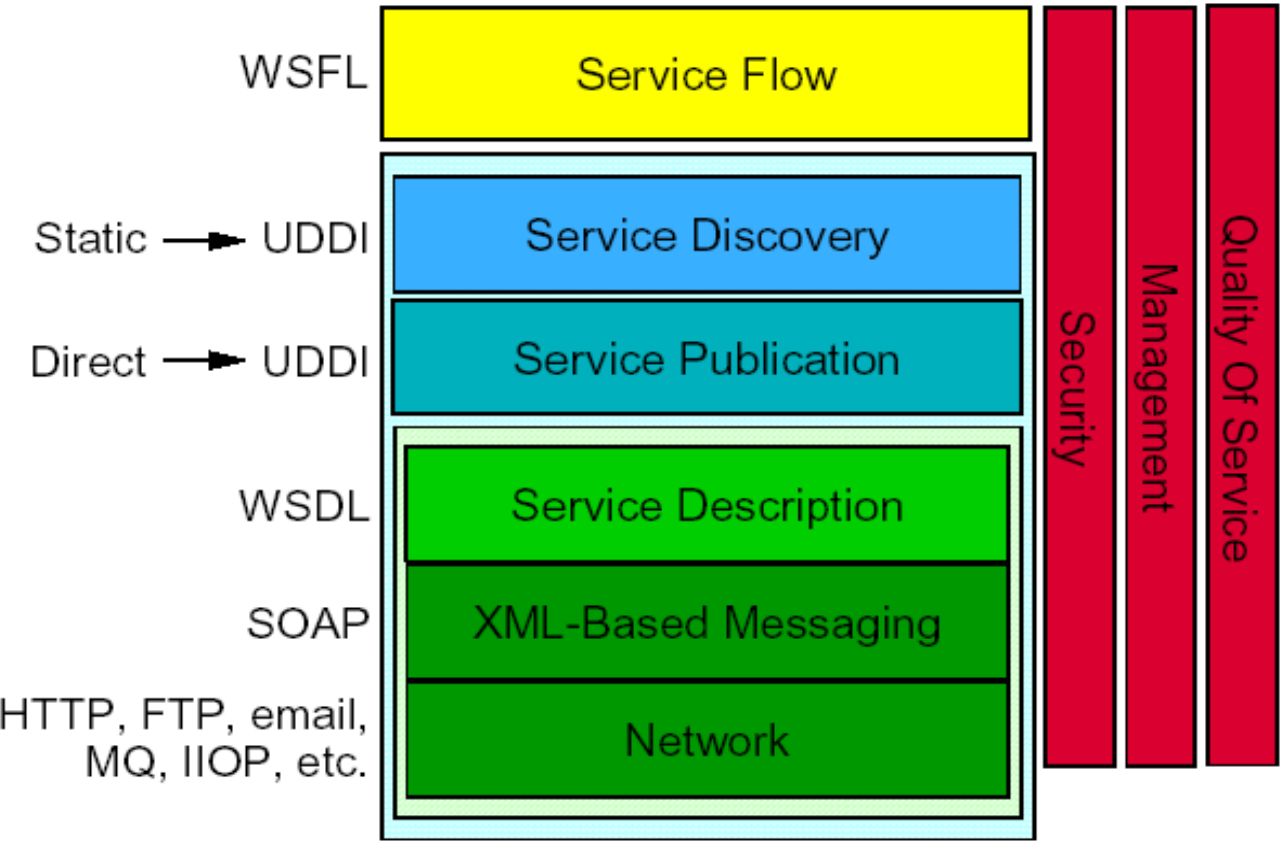
Web Service Architectures

- Allow data on Web to be accessed using remote procedure call mechanism
- Two approaches are widely used
  - **Big Web Services:**
    - built-on top of HTTP
    - use XML representation for sending request data, as well as for returning results following the Simple Object Access Protocol (SOAP) standard
      - XML Messaging Protocol
    - define interfaces syntactically written in the Web Services Description Language (WSDL) standard
      - 서비스 제공 장소, 서비스 메시지 포맷, 프로토콜 등을 기술함
    - **SOA (Service Oriented Architecture) 라고 함**
  - **REST (Representation State Transfer) Service 방식 제안 (2000년)**
    - **Big Web Service가 Web의 장점들은 살리지 못한 다는 비판**
      - **XML SOAP 기반 Remote Procedure Call의 overhead**
    - HTTP의 기본 기능만으로 원격 정보에 접속 하자는 의도

The Big Web Service Stack

참고자료

WSDL: Web Service Description Language with XML
UDDI: Universal Description, Discovery and Integration
WSFL: Web Service Flow Language



- SOAP is a **W3C standard** XML based **communication protocol**
  - a format for **sending messages between applications via Internet**
  - **platform independent**
  - **language independent**
  - **simple and extensible**

SOAP Request/Response Message

참고자료

```
<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.stock.org/stock">

   <m:GetStockPrice>
     <m:StockName>IBM</m:StockName>
   </m:GetStockPrice>

  </soap:Body>
</soap:Envelope>
```

SOAP Envelope Namespace

Message

Message Namespace

SOAP Envelope

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
     <soap:Body xmlns:m="http://www.stock.org/stock">
      <m:GetStockPriceResponse>
        <m:Price>34.5</m:Price>
      </m:GetStockPriceResponse>
     </soap:Body>
</soap:Envelope>
```

Result returned in Body

Message

SOAP Envelope

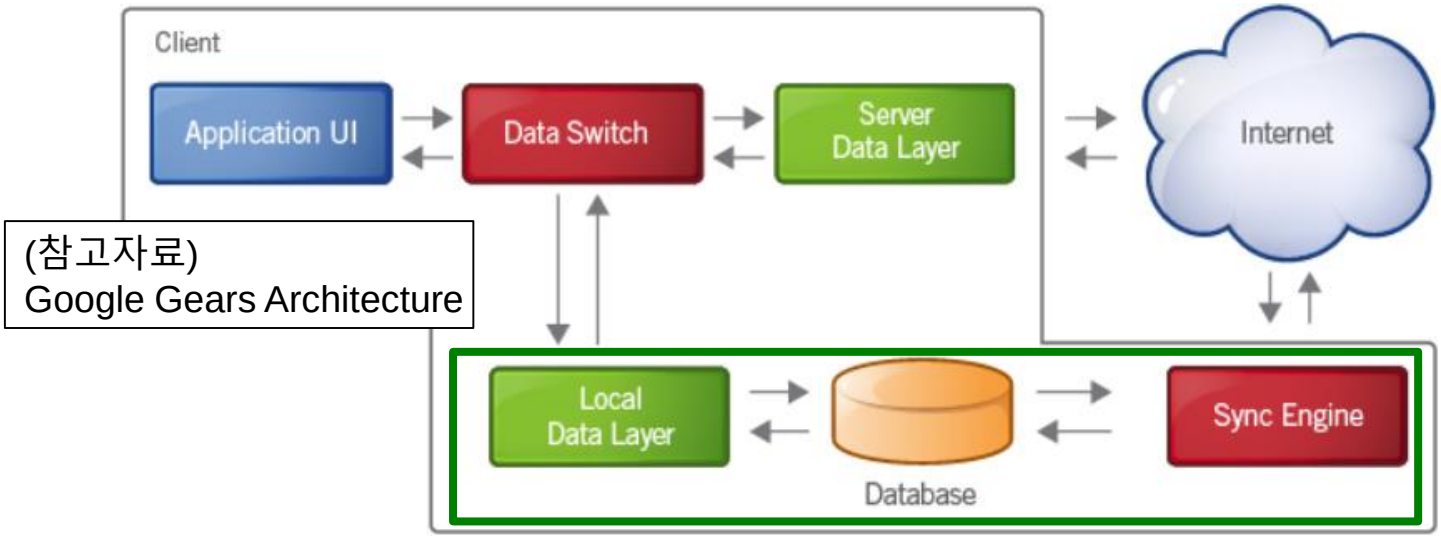
REST Architecture for Web Service

- **RESTful Web Services: using Representation State Transfer (REST)**
  - **Web의 장점을 최대한 살리는 Web Service Architecture (2000년에 제안)**
  - is well suited for basic, ad hoc integration scenarios
  - allows use of standard HTTP request to a URL to execute a request and return data
    - HTTP URI + HTTP Method만 가지고 Web Resource와 Web Resource에 대한 action을 명시하는것으로 Web Service를 실현
  - returned data is either in various data formats
    - XML
    - **JavaScript Object Notation (JSON)**
      - 속성-값 쌍으로 이루어진 데이터를 전달하기 위해 사용하는 개방형 표준 포맷
      - 비동기 브라우저/서버 통신에서 XML을 대체하는 포맷

- Big Web Service의 표준체계가 복잡하여 Google, Amazon같은 진영에서 가벼운 Web Service Architecture를 원하는것에 힘입어서 최근에 활발하게 사용됨
  - 그러나 표준이 없다보니 관리도 어려운점이 상존

Client Operation Running in a Disconnected State

- Tools for applications to use the Web when connected, but operate locally when disconnected from the Web
- Even in disconnected state, parallel execution of Client SW and Server SW must be provided
  - Google Gears (a browser plugin)
    - Provide a local database, a local Web server and support for execution of JavaScript at the client
    - JavaScript code using Gears can function identically on any OS/browser platform
  - Adobe AIR software provides similar functionality outside of Web browser



(참고자료)
Google Gears Architecture

Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Rapid Application Development

- A lot of effort is required to develop modern Web application interfaces
- Web 2.0
  - 웹 어플리케이션을 제공하는 완전한 플랫폼으로서의 웹
  - 사용자끼리의 상호작용, 사용자 중심의 정보 공유를 추구하는 웹
  - 사례) UCC, Service Wikis, Social Network Individual

- Several approaches to speed up application development
  - Function library to generate user-interface elements
  - Drag-and-drop features in an IDE to create user-interface elements
  - Automatically generate code for user interface from a declarative specification

- Above features have been in used as part of **rapid application development (RAD)** tools even before advent of Web

- GUI Building Tools
- Web Application Frameworks

GUI Building Tools

- JSP (Java Server Page) / Java
- MS Active Server Page (ASP)
- MS ASP.NET
  - Dynamic Web Page, Web App, Web Service를 생성하기 위한 Web Application Framework
  - HTML안에 Visual Basic이나 C# 삽입
  - ASP는 3.0 버전 이후에 ASP.NET로 대체
- MS Visual Studio provides drag-and-drop development using these controls below
  - Menus and list boxes can be associated with DataSet object
  - Validator controls (constraints) can be added to form input fields
    - JavaScript to enforce constraints at client, and separately enforced at server
  - User actions in client GUI can be associated with actions at server
  - DataGrid displays SQL query results in tabular format (HTML 생성)

- Java Server Faces (JSF) framework
  - includes JSP tag library: JSP 태그 라이브러리를 이용한 사용자 인터페이스 작성 가능

Web Application Frameworks

- Framework ➔ Automatic Code Generation
- Web Application Framework은 일반적인 framework보다 좀더 web development 쪽
- Major Common Features
  - OO Data Model with Object-Relational Mapping (ORM)
    - Simple CRUD Interfaces (create, read, update and delete of objects/tuples) by code generation from database schema or object model
  - Declarative way of specifying a form with validation constraints on your input
    - Javascript/Ajax
  - Template script system (similar to JSP)

- Ruby 기반
  - Ruby on Trail
- Java/JSP 기반
  - Jboss Seam, Apache Struts, Swing, Tapestry, WebObjects

Report Generator for A Formatted Report

- Tools for generating human-readable summary reports from a database
  - Report Generator는 query결과를 텍스트 및 도표로 통합하여 보여줌
- Cristal Reports by Business Objects (now by SAP)
  - Chart generation facilities
- MS SQL Server Reporting Services
  - Loading query results into Excel

Acme Supply Company, Inc.
Quarterly Sales Report

Period: Jan. 1 to March 31, 2009

Region	Category	Sales	Subtotal
North	Computer Hardware	1,000,000	
	Computer Software	500,000	
	All categories		1,500,000
South	Computer Hardware	200,000	
	Computer Software	400,000	
	All categories		600,000
		Total Sales	2,100,000

참고자료

Many Many Reporting Tools

- Cristal Reports by Business Objects (now by SAP)
- MS SQL Server Reporting Services
- Oracle Reports
- MicroStrategy
- Tibco SpotFire
- Splunk

- Sometimes called, BI Tools
- **Business intelligence** (**BI**) can be described as "a set of techniques and tools for the acquisition and transformation of raw data into meaningful and useful information for business analysis purposes and effective decision making“
- BI technologies
  - provide historical, current and predictive views of business operations
  - reporting, OLAP, analytics, data mining, business performance management, benchmarking, text mining, predictive analytics and prescriptive analytics

참고자료

Tibco SPOTFIRE

Spotfire는 사용자 중심의 직관적이고 사용하기 용이한 UI/UX를 갖고 있으며 각종 DB, DW, local file 등을 한 화면 내에서 분석 할 수 있습니다. 또한 R, SAS 등 외부 통계 툴과 연계하여 고급 분석을 실행 할 수 있으며 분석 결과를 파일 형태로 뿐만 아니라 웹이나 모바일 기기로 공유 가능 하여 기업 내 다양한 사용자와 대량 사용자에 의한 접근에 최적화된 Enterprise solution입니다.

SPLUNK

- Captures, indexes and correlates real-time data in a searchable repository
- Generate graphs, reports, alerts, dashboards and visualizations

Chapter 9: Application Design and Development

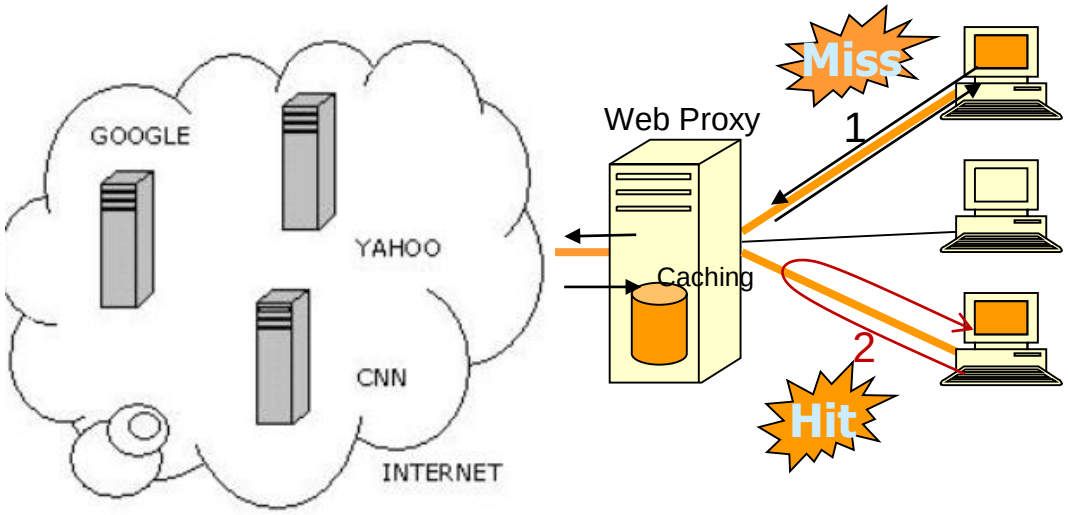
- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Improving Web Server Performance by Caching

- Performance is an issue for popular Web sites
  - May be accessed by millions of users every day, thousands of requests per second at peak time
- Caching techniques used to reduce cost of serving pages by exploiting commonalities between requests
  - At the server site:
    - Caching of JDBC connections
      - Connection pooling between servlet requests
    - Caching query results
      - Cached results must be updated if underlying database changes
    - Caching of generated HTML
  - At the client's network
    - Caching of pages by **Web proxy**

Clientside Caching by Web Proxy

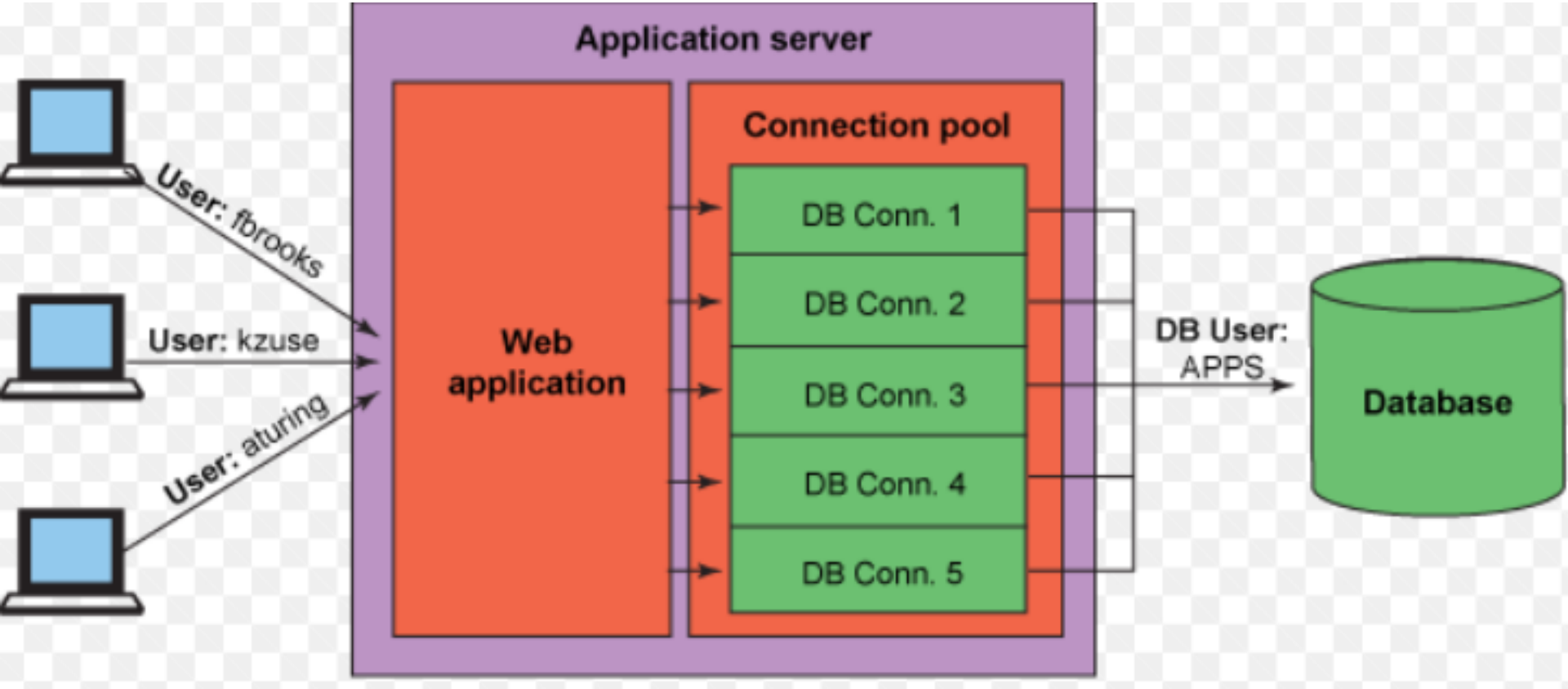
참고자료



- 프록시 서버는 자주 요청된 페이지들을 로컬에 캐싱해둠으로써 접근성능을 향상
- 프록시서버는 외부와의 트래픽을 줄여서 네트워크 병목 현상을 방지

참고자료

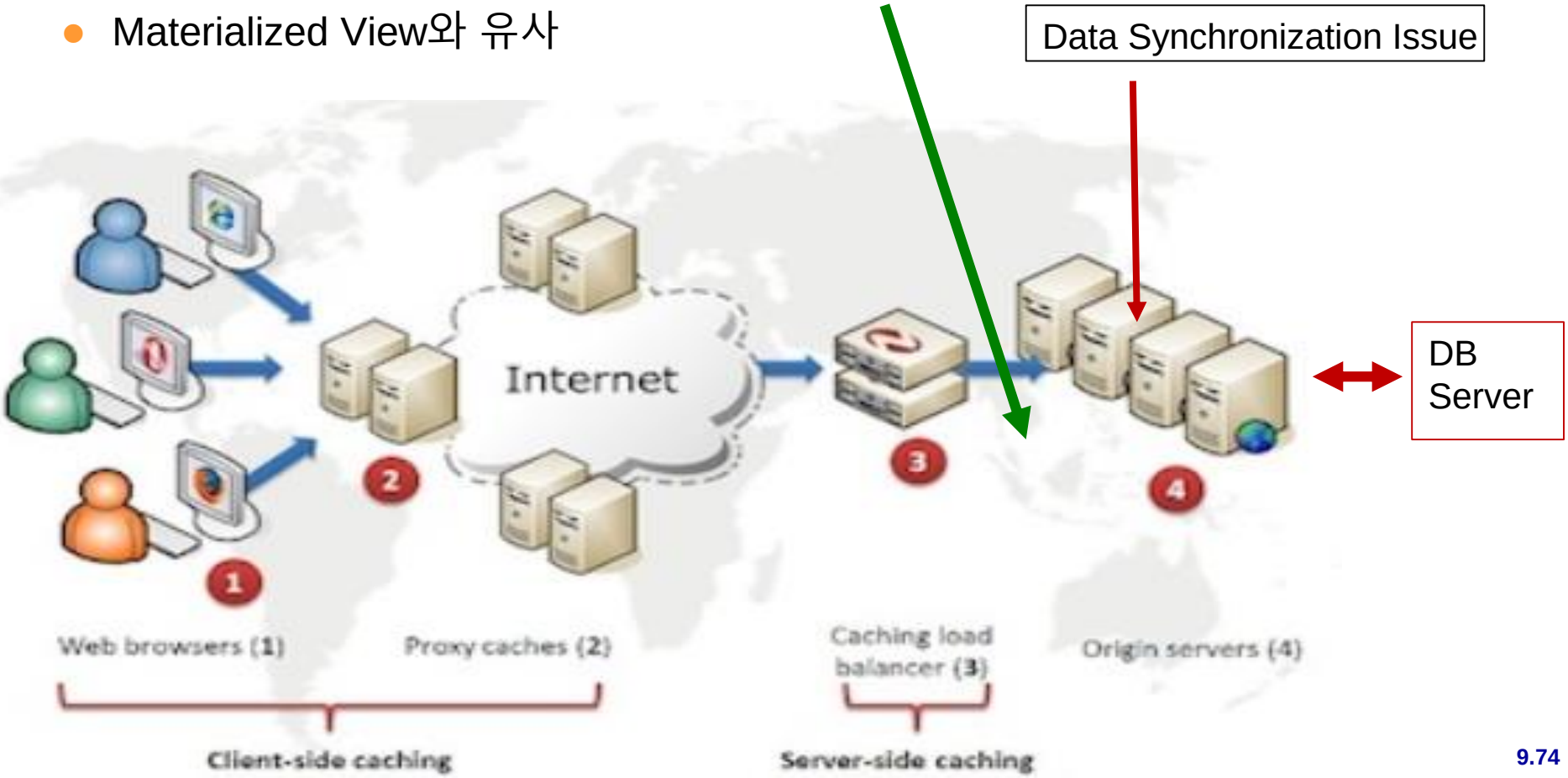
Serverside Caching by Connection Pooling



- DB접근 사용자 요청이 들어올 때 마다 새로운 JDBC 커넥션을 만드는 것은 최소 수 밀리세컨이 걸리기 때문에 사용자가 매우 많을 때는 전체 시스템의 성능을 지연
- 새로운 커넥션을 생성하는 대신에 connection pool로부터 사용 가능한 커넥션을 할당 받는 방법

Improving Web Server Performance by Parallel Application Servers

- Use a large number of application servers in parallel
- The underlying database is shared by all the application servers
- Bottleneck: the database server
- Caching query results at the application server
  - Materialized View와 유사



Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

SQL Injection Attack

- Suppose query is constructed using
  - "select * from instructor where name = '" + name + "'"
- Suppose the user, instead of entering a name, enters:
  - X' or 'Y' = 'Y
- then the resulting statement becomes:
  - "select * from instructor where name = '" + "X' or 'Y' = 'Y" + "'"
  - which is:
    - select * from instructor where name = 'X' or 'Y' = 'Y'
  - User could have even used
    - X'; update instructor set salary = salary + 10000; --

Where절이 항상 참이므로
전체 instructor 릴레이션이 반환됨

한번의 호출로 여러 개의 구문을
허용하는 오래된 시스템의 경우

- JDBC Prepared statement internally uses:
  "select * from instructor where name = 'X\' or \'Y\' = \'Y'

이러한 이름을 지닌 교수를 반환(empty)

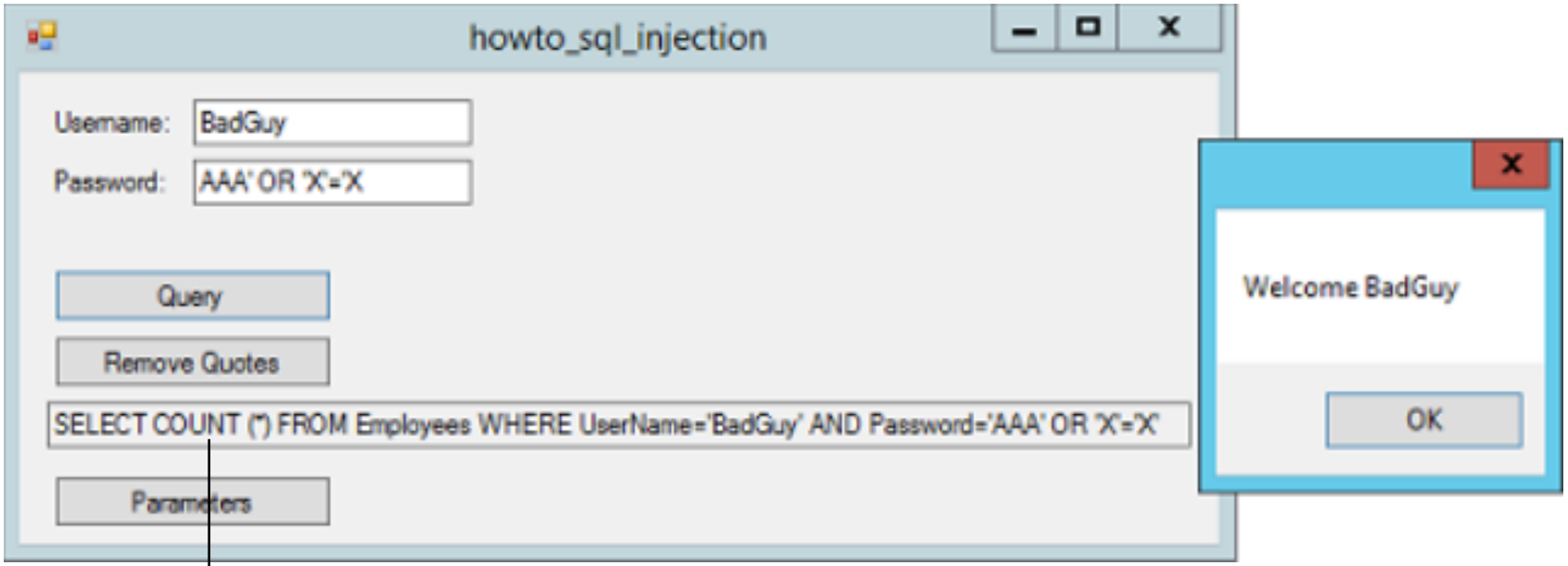
- **Always use prepared statements, with user inputs as parameters**
  - conn.prepareStatement("select * from instructor where name = '" + name + "'")

Prepared statement는 입력할 값들을 ?로 대체하여 pre-compile하고, 나중에 ?에 값을 대입한후 질의를 수행하는 방식. ?에 값을 대입하는 setString 메소드는 quote를 escape하도록 구현.
위 예제에 X or Y = Y 라는 이름을 지닌 교수를 추출하면 빈 결과가 반환되어짐

SQL Injection Attack Example

참고자료

SQL String을 만들어서 JDBC를 이용하여 (Prepared Statement 쓰지 않고) DB Server로 보냈다면



일반적으로 Select count(*) 결과가 0이면 로그인을 허가하지 않고, Select count(*) 결과가 1인 경우 로그인을 허용하도록 구현되어 있음. 이러한 경우 Injection Attack에 의해 Select count(*) 결과가 무조건 1이 되므로 로그인이 허가됨

Cross Site Scripting(XSS) and Cross-Site Request Forgery (CSRF)

- HTML code on one page executes action on another page
  - 이름이나 댓글을 입력하는 대신에 JavaScript나 Flash같은 Script언어를 입력해두고, 사용자가 그 입력을 클릭하면 Script를 실행하여 원치 않는 작업을 수행
- XSS: Script를 click하면 개인적인 Cookie를 넘기게 하는 공격
- CSRF: Script를 click하면 해커가 만들어 놓은 web site에서 작업을 하게하는 공격

E.g. <img src = http://mybank.com/transfermoney?amount=1000&toaccount=14523>

  - Risk: if user viewing page with above code is currently logged into mybank, the transfer may succeed
  - Above example simplistic, since GET method is normally not used for updates, but if the code were instead a script, it could execute POST methods
- **Prevent your web site from being used to launch XSS or CSRF attacks**
  - Disallow HTML tags in text input provided by users, using functions to detect and strip such tags (HTML Tag 입력금지)
- **Protect your web site from XSS/XSRF attacks launched from other sites**
  - ..next slide

참고자료

Protecting XSS/XSRF attacks

- **Protect your web site from XSS/XSRF attacks launched from other sites**
  - Use **referer** value (URL of page from where a link was clicked) provided by the HTTP protocol, to check that the link was followed from a valid page served from same site, not another site
  - Ensure IP of request is same as IP from where the user was authenticated
    - prevents hijacking of cookie by malicious user
  - Never use a GET method to perform any updates (use POST method)
    - This is actually recommended by HTTP standard

URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Request Headers | Request Body | Response Headers | Response Body | Cookies | Initiator | Timings

Key	Value
Request	GET /Protocols/rfc2616/rfc2616-sec14.html HTTP/1.1
Accept	text/html, application/xhtml+xml, */*
Referer	http://www.google.com/url?sa=t&source=web&cd=3&ved=0CC4QFjAC
Accept-Language	en-US
User-Agent	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5
Accept-Encoding	gzip, deflate
Host	www.w3.org
If-Modified-Since	Wed, 01 Sep 2004 13:24:52 GMT
If-None-Match	"1edec-3e3073913b100"
Connection	Keep-Alive

HTTP request 헤더를 시각화해서 보여준 예

이전 슬라이드의 <img src..> 같은 공격을 막을 수 있음

Password Leakage

- Never store passwords, such as database passwords, in clear text in scripts that may be accessible to users
  - 암호화를 시켜서 저장해야 함

- Restrict access to database server from IPs of machines running application servers
  - Most DBMSs allow restriction of access by source IP address
  - 은행에 login을 하는데 특정 IP Address가 아님 다른 IP Address에서 접속하면 거부

Application Authentication [1/3]

- Single factor authentication such as passwords too risky for critical applications
  - guessing of passwords, sniffing of packets if passwords are not encrypted
    - Sniff: 훔쳐보다
  - passwords reused by user across sites
  - spyware which captures password

- Two-factor authentication scheme
  - password plus one-time password devices (such as OTP device)
    - device generates a new pseudo-random number every minute, and displays to user
    - user enters the current number as password
    - application server generates same sequence of pseudo-random numbers to check that the number is correct
  - password plus one-time password sent by SMS (short message system)

Application Authentication [2/3]

- **Man-in-the-middle** attack
  - Web site (bogus site) that pretends to be mybank.com, and passes on requests from user to mybank.com, and passes results back to user
  - Even two-factor authentication cannot prevent such attacks
  - Solution: authenticate Web site to user, using digital certificates, along with secure http protocol
    - HTTPS 프로토콜은 사용자가 가짜 웹사이트로 연결하지 않도록 웹사이트를 인증하는 기능을 포함하고 있음

- **Central authentication** within an organization
  - application redirects to central authentication service for authentication
  - avoids multiplicity of sites having access to user's password
  - LDAP or Active Directory used for central authentication

Application Authentication [3/3]

- **Single sign-on system** allows user to be authenticated once, and applications can communicate with authentication service to verify user's identity without repeatedly entering passwords
  - **Security Assertion Markup Language (SAML)** standard for exchanging authentication and authorization information across security domains
    - User from Yale signs on to external application such as acm.org using userid joe@yale.edu
    - Application communicates with Web-based authentication service at Yale to authenticate user, and find what the user is authorized to do by Yale (e.g. access certain journals)
    - 기관 사이에 인증을 교환하기 위한 표준
  - **OpenID** standard allows sharing of authentication across organizations
    - Application allows user to choose Yahoo! as OpenID authentication provider, and redirects user to Yahoo! for authentication
    - Portal이 인증 제공자 역할을 함

SQL-Level Authorization [1/2]

- Current SQL standard does not allow fine-grained authorization such as "students can see their own grades, but not other's grades"
  - Problem 1: Database is lack of end-user information
    - 보통 user class에 대해서 authorization이 만들어짐
  - Problem 2: SQL authorization is at the level of tables, or columns of tables, but not to specific rows of a table

- One workaround: use views such as

```
create view  studentTakes  as
select *
from   takes
where takes.ID = syscontext.user_id()
```

  - where syscontext.user_id() provides end user identity
    - end user identity must be provided to the database by the application
  - Having multiple such views is cumbersome
    - 모든 학생 A, B, C, ….Z 에 대해서 개인별 view를 만들어야 하는데…
    - "학생개인은 자기성적만 볼수 있다"를 표현하길 원함

SQL-Level Authorization [2/2]

- Currently, authorization is done entirely in database application
- Entire application code has access to entire database
  - large surface area, making protection harder
  - 예. Application에서 부주의로 인해 권한 검사를 놓치게 될 경우, 권한 없는 사용자가 기밀 자료에 접근 할 위험성이 존재하게 됨

- Alternative: **fine-grained (row-level) authorization** schemes
  - extensions to SQL authorization proposed but not currently implemented
  - Oracle Virtual Private Database (VPD) allows predicates to be added transparently to all SQL queries, to enforce fine-grained authorization
    - e.g. add *ID= sys_context.user_id()* to all queries on student relation if user is a student
    - row-level authorization = tuple-level authorization
  - Side effect: 예를 들어 모든 수업의 평균 학점을 보고 싶은데, 강제적인 row-level 인증으로 인해 사용자의 평균 학점만이 결과로 나옴

Audit Trails

- A log of the followings
  - all changes to the application data
  - which user performed the change
  - when the change was performed

- Applications must log actions to an audit trail, to detect who carried out an update, or accessed some sensitive data
  - DB Transaction Log <transaction_ID, variable, old_value, new_value, time>
    - Database-level Audit Trail
  - Application-level Audit Trail (what action, by whom, by when, from which IP)

- Application-level Audit Trails used after-the-fact (사후에) to
  - detect security breaches
  - repair damage caused by security breach
  - trace who carried out the breach

참고자료

Audit Trails Example

ZendServer : PHP application Server + Support web application frameworks

Audit Trail에 화면: IP, 권한, 날짜, 작업내용 등의 로그를 저장하고 있는 것을 볼 수 있음

zend Server ENTERPRISE | Overview | Applications | Configurations | Administration | administrator | 10:33

Servers | Audit Trail | Users | WebAPI | License | Settings | Import/Export

Administration > Audit Trail

Show Filters | All Audit Entries | Time Range: All | From: 13/Jan/2013 10:32 | To: 14/Jan/2013 10:32

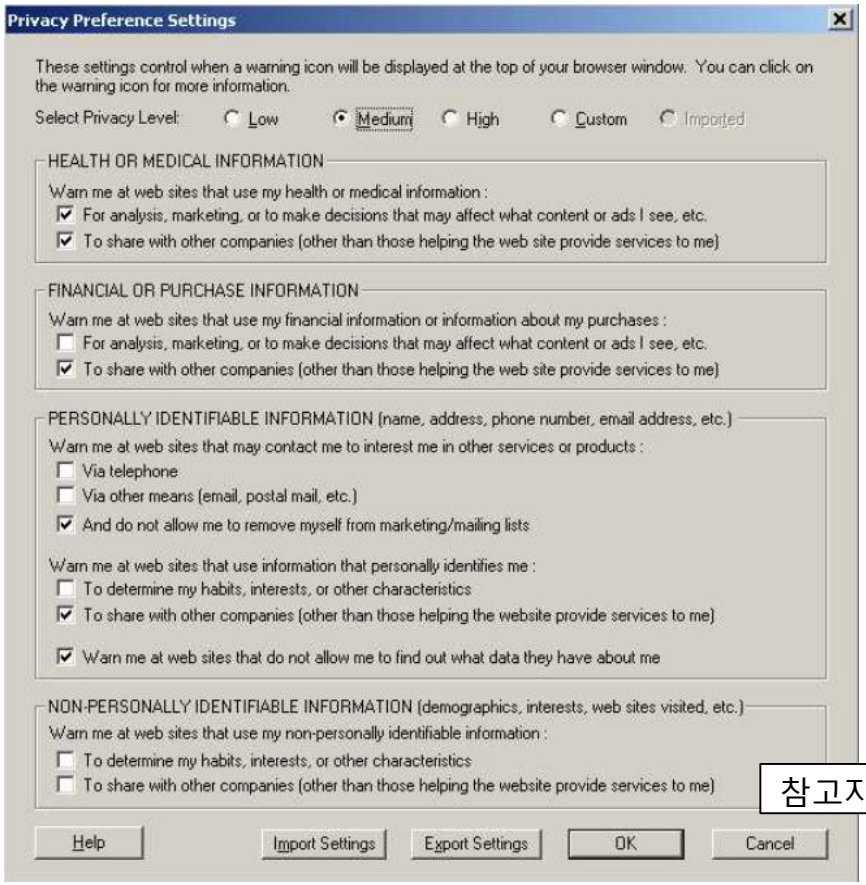
Filter By: Currently no filter selected

Shown: 1-12 / 12 | Export

ID	Origin	User	Creation Time	Operation	Outcome	Extra Details
12	127.0.0.1 (GUI)	admin	Today, 10:32:44	Job Queue rule run	OK	ruleId=1
11	127.0.0.1 (GUI)	admin	Today, 10:32:36	Job Queue rule saved	OK	name=MyJob, schedule=1 */2, url=http://localhost/...
10	127.0.0.1 (GUI)	admin	Today, 10:32:09	Application defined	OK	
9	127.0.0.1 (GUI)	admin	Today, 10:26:30	PHP restarted	OK	
8	127.0.0.1 (GUI)	admin	Today, 10:26:30	New license stored	OK	
7	127.0.0.1 (GUI)	admin	Today, 10:25:59	GUI authenticated	OK	
6	127.0.0.1 (GUI)	admin	Today, 10:25:54	GUI authenticated	Failed	wrong username password combination
5	127.0.0.1 (WebAPI)	System Key	Today, 10:21:18	Page Cache Save Rule	OK	name=time, ruleId=-1
4	127.0.0.1 (WebAPI)	System Key	Today, 10:21:17	Monitor rule added	OK	Monitor rule name: PHP Error
3	127.0.0.1 (WebAPI)	System Key	Today, 10:21:17	Monitor rule added	OK	Monitor rule name: Custom Event
2	127.0.0.1 (GUI)	admin	Today, 10:21:14	Application deployed	OK	There are 2 messages
1	127.0.0.1 (GUI)	Unknown	Today, 10:07:36	Bootstrap license saved	OK	

Privacy in Database

- Database having personal data
- Medical data: 개인 정보를 비식별화 시켜서 프라이버시를 유지한 자료로 연구
- Purchase data
- Many web sites allow customers to specify their privacy preferences



참고자료

한국의 많은 site들은 동의를 안하면 그다음으로 넘어가지 않도록 되어 있는 경우가 많음

Chapter 9: Application Design and Development

- 9.1 Application Programs and User Interfaces
- 9.2 Web Fundamentals
- 9.3 Servlets and JSP
- 9.4 Application Architectures
- 9.5 Rapid Application Development
- 9.6 Application Performance
- 9.7 Application Security
- 9.8 Encryption and Its Applications

Encryption [1/3]

- Data may be *encrypted* when database authorization provisions do not offer sufficient protection.
- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - Encryption scheme depends not on the secrecy of the algorithm but on the secrecy of a parameter of the algorithm called the encryption key.
  - Extremely difficult for an intruder to determine the encryption key.
- **Symmetric-key encryption**: same key used for encryption and for decryption
- **Public-key encryption** (a.k.a. **asymmentric-key encryption**): use different keys for encryption and decryption
  - encryption key can be public, decryption key secret

Encryption [2/3]

- *Data Encryption Standard* (DES)
  - substitutes characters and rearranges their order on the basis of an encryption key which is provided to authorized users via a secure mechanism
  - Scheme is no more secure than the key transmission mechanism since the key has to be shared
- Advanced Encryption Standard (AES) is a new standard replacing DES, and is based on the Rijndael algorithm, but is also dependent on shared secret keys
- *Public-key encryption* is based on each user having two keys:
  - *public key* – publicly published key used to encrypt data, but cannot be used to decrypt data
  - *private key* -- key known only to individual user, and used to decrypt data. Need not be transmitted to the site doing encryption.

  Encryption scheme is such that it is impossible or extremely hard to decrypt data given only the public key.
- The RSA public-key encryption scheme is based on the hardness of factoring a very large number (100's of digits) into its prime components

참고자료

Encryption Algorithms

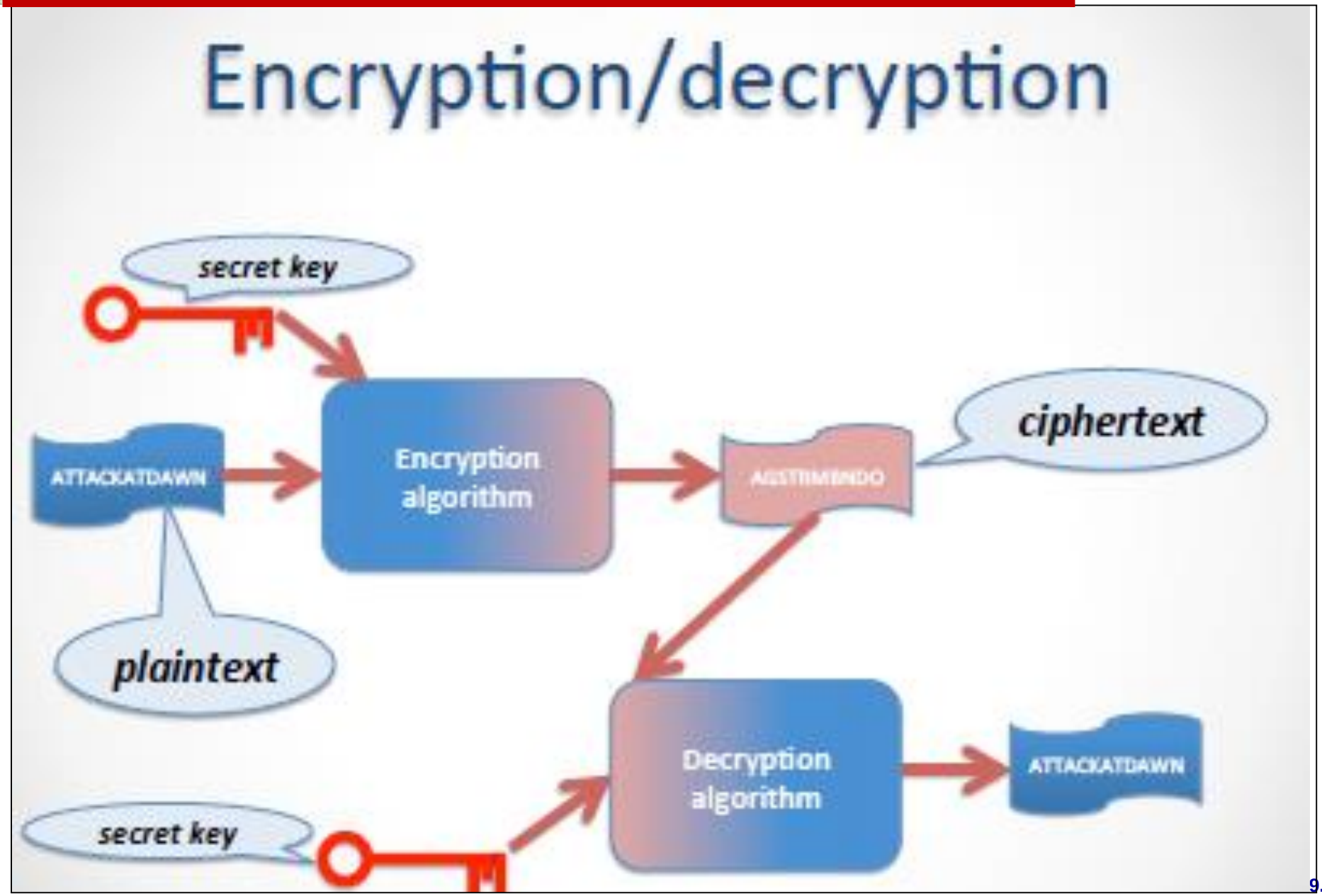
- DES(Data Encryption Standard)
  - 미국 표준기술협회인 NIST가 1977년에 발표한 암호방식
  - 64bit Key와 개별 키 암호화 방식을 통해 텍스트를 암호화
  - 메시지는 64비트블록으로 나뉜뒤 데이터 암호화 알고리즘을 통해 암호화
  - 개별 키 암호화 방식에서는 발신자과 수신자만이 키를 공유
  - 키를 모르면 코드를 풀기가 어렵지만, 해당키의 보안유지는 사용자의 몫
  - 신용카드에 많이 사용
- AES(Advanced Encryption Standard)
  - 2000년 미국 정부에 의해서 표준화된 대칭키 암호화 알고리즘
  - Rijndael 알고리즘을 기반으로 함
  - 암호화 키로 128, 192, 256 비트의 가변키 지원
- RSA
  - 1978년 Ron Rivest, Adi Shamir, Leonard Adleman의 연구에 의해 체계화됨
  - 비대칭키 암호화 방식(공개키 암호화시스템)
  - 전자서명이 가능한 최초의 알고리즘

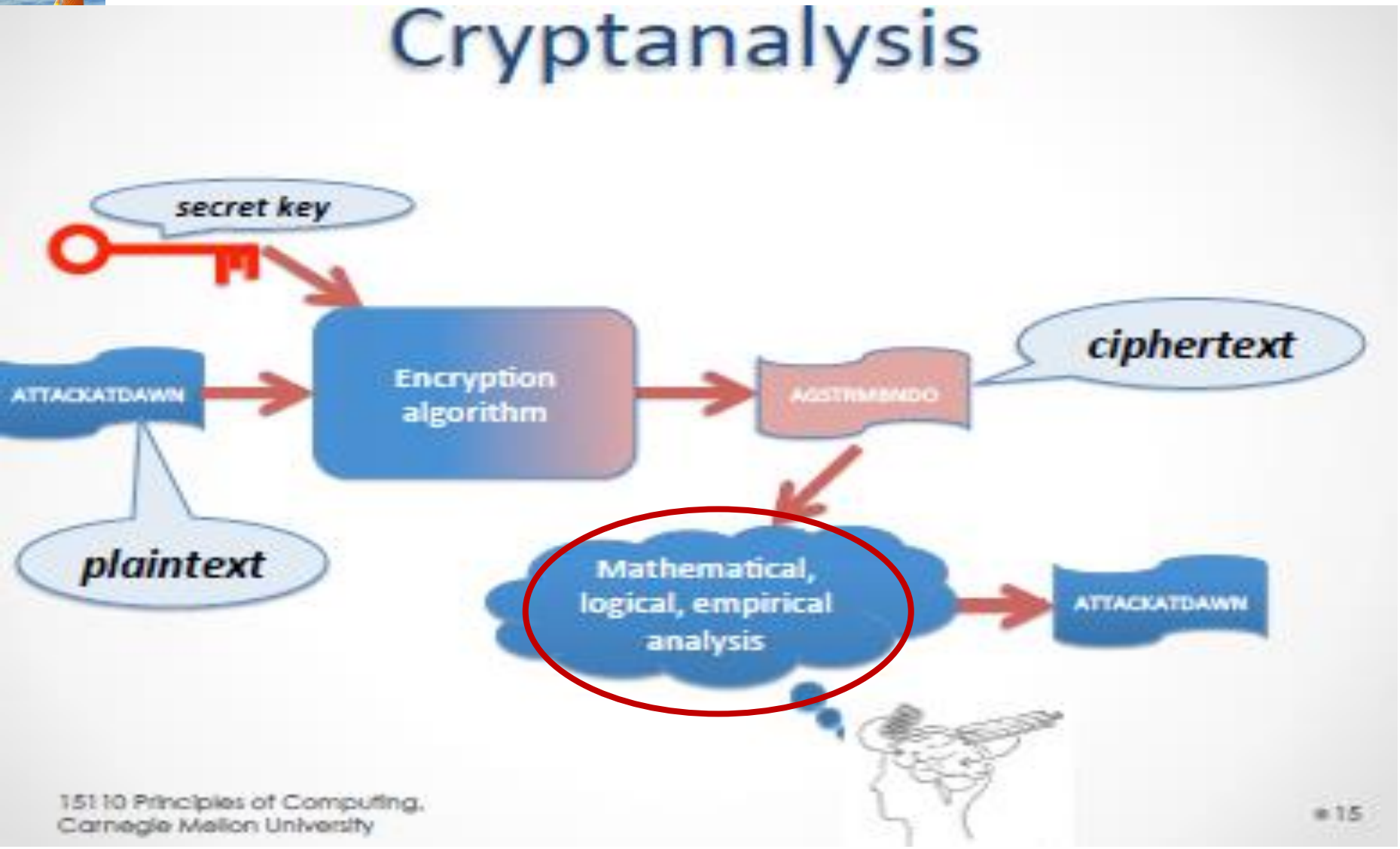
Encryption [3/3]

- **Hybrid schemes** for a large amount of data
  - Public encryption key (PEK) is computationally expensive
  - A symmetric encryption key SEK is randomly generated
  - The key from SEK is transmitted with PEK safely
  - Then, SEK is used for transmitting a large data amount of data

- **Dictionary attacks: 쉽게 생산가능한 조합으로 공격하는 방법**
  - Data_of_birth, Zip code, etc
  - but even otherwise, statistical information such as frequency of occurrence can be used to reveal content of encrypted data
  - Can be deterred by adding extra random bits to the end of the value, before encryption, and removing them after decryption
    - same value will have different encrypted forms each time it is encrypted, preventing both above attacks
    - extra bits are called **salt bits or initialization vector**
    - **Extra random bits을 모르면 Date_of_birth를 다 생산해서 비교해도 같은값을 만들수 없다**

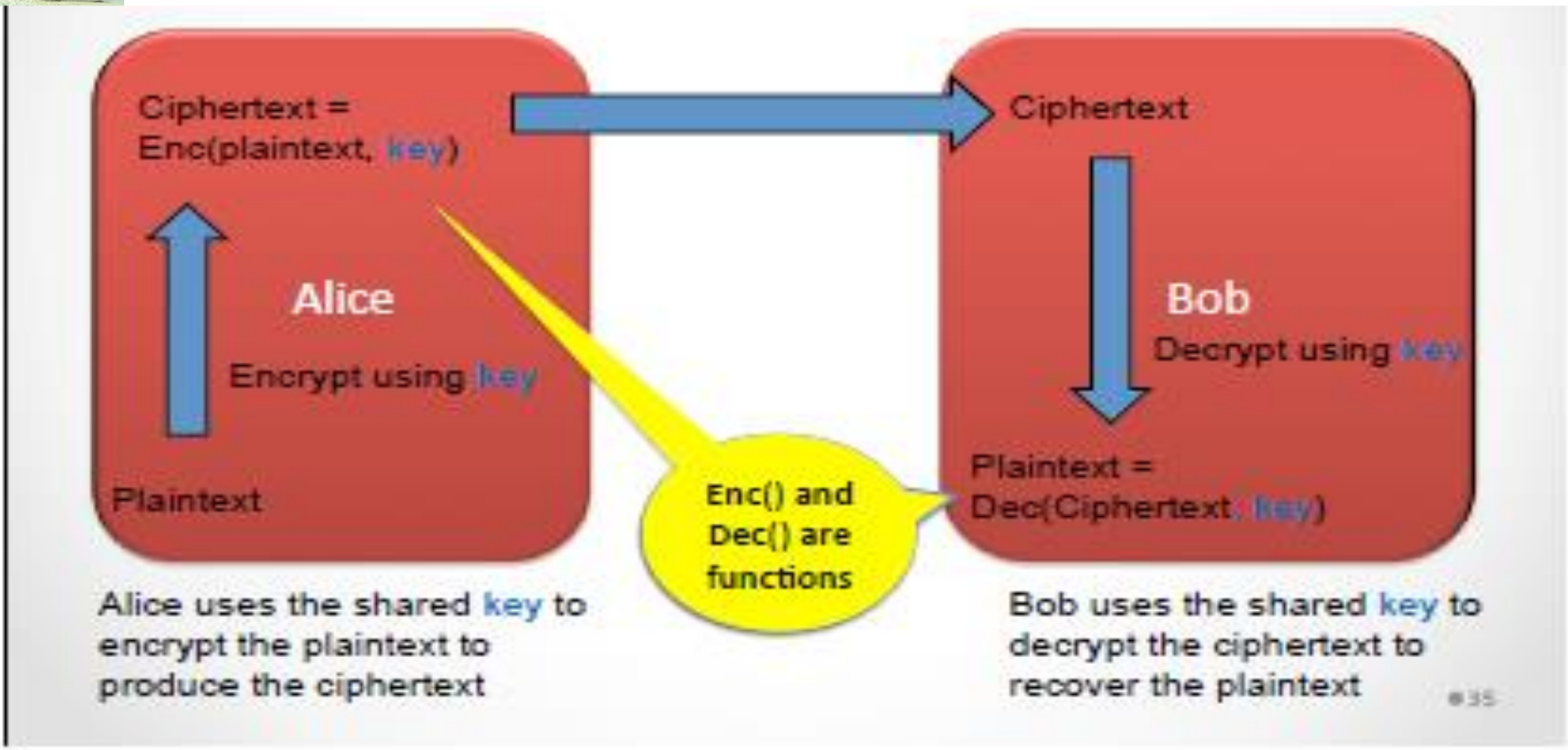
Encryption in computing





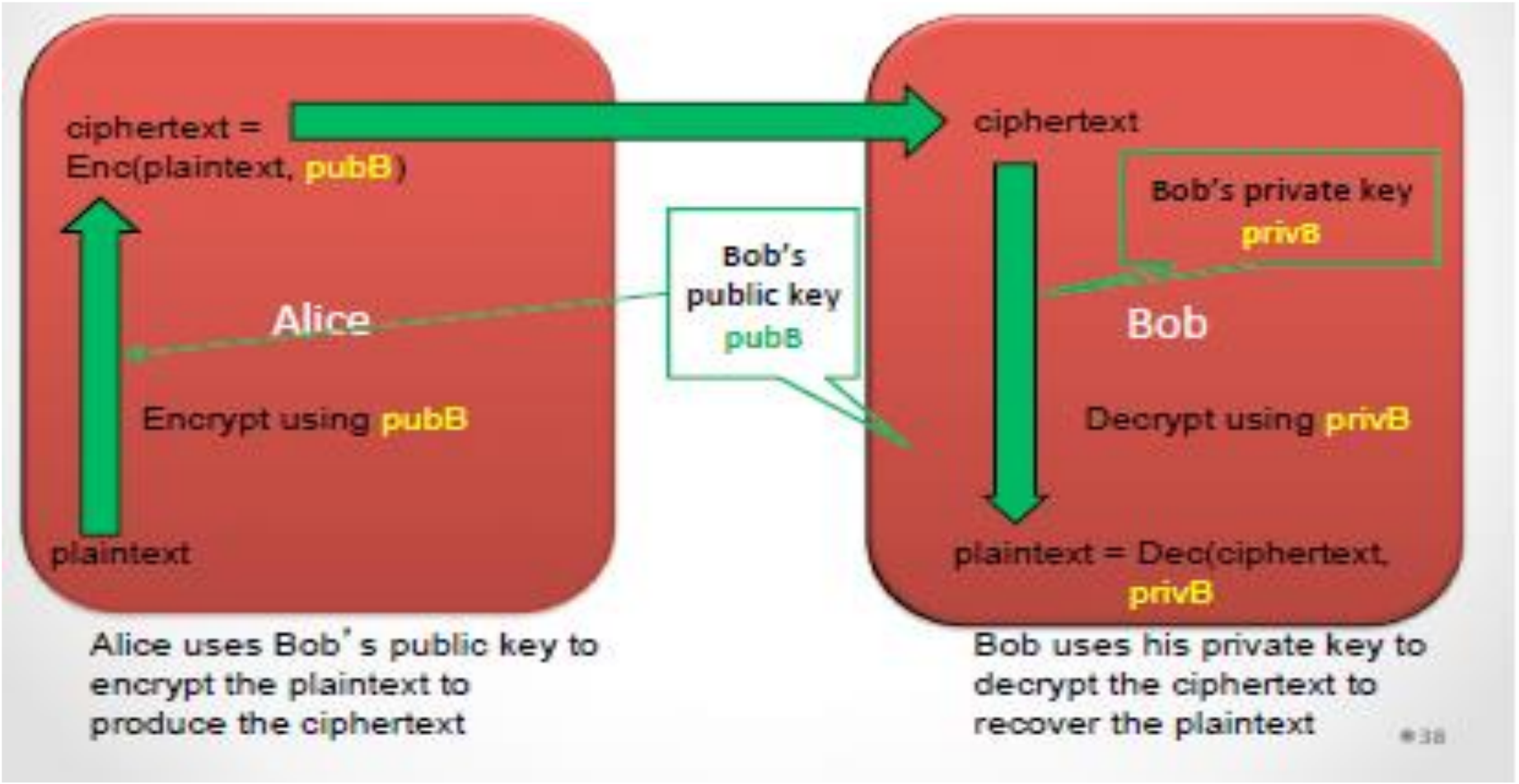
Cryptanalysis (크립태넬러시스) = 암호해독

Symmetric Encryption using Shared Key



- Problem: How Alice and Bob agree on a secret key? Using a public communication system?
- Solution: asymmetric encryption based on the number theory
  - Alice has one secret key, Bob has a different secret key, working together they established one shared secret key
  - RSA Public Key Encryption, Diffie-Hellman Key Exchange

Asymmetric Public Key Encryption



One Type of Asymmetric Encryption: RSA

- Named after its inventors: Rivest, Shamir and Adleman
- Used in https, ssl/tls

- First, we must be able to represent any message as a single number (it may already be a number as is usual for a symmetric key)
  - For example:

    A T T A C K A T D A W N

    01202001031101200401231 4

- Public Key and Private Key

  - Every receiver has a public key (e, n) and a private key (d, n). (used for encryption / used for decryption)
  - The transmitter encrypts a (numerical) message M into ciphertext C using the receiver's public key:

    M^e modulo $n \longrightarrow C$ *(ciphertext)*
  - The receiver decodes the encrypted message C to get the original message M using the private key (which no one else knows).

    C^d modulo $n \longrightarrow M$ *(plaintext)*

RSA Example

- Alice's Public Key: (3, 33) (e = 3, n = 33)
- Alice's Private Key: (7, 33) (d = 7, n = 33)
  - Usually these are really huge numbers with many hundreds of digits!
- Bob wants to send the message 4
  - Bob encrypts the message using *e* and *n*:
    4^3 modulo 33 → 31 ... Bob sends 31
- Alice receives the encoded message 31
  - Alice decrypts the message using d and n:
    31^7 modulo 33 → 4

Generating n, e and d

- p and q are (big) random primes. — $p = 3, q = 11$
- $n = p \times q$ — $n = 3 \times 11 = 33$
- $\varphi = (p - 1)(q - 1)$ — $\varphi = 2 \times 10 = 20$
- e is small and relatively prime to φ — $e = 3$
- d, such that: $e \times d \bmod \varphi = 1$ — $3 \times d \bmod 20 = 1$, $d = 7$

Usually the primes are huge numbers--hundreds of digits long.

Cracking RSA

- Everyone knows (e, n). Only Alice knows d.
- If we know e and n, can we figure out d?
  - If so, we can read secret messages to Alice.
- We can determine d from e and n.
  - Factor n into p and q.

    $n = p \times q$

    $\varphi = (p - 1)(q - 1)$

    $e \times d = 1 \pmod{\varphi}$
  - We know e (which is public), so we can solve for d.
- But only if we can factor n

e, n 값은 알려진것이고 n을 break해서 p, q만 알아내면,
φ을 알고 그러면 d값을 알아낼수 있다!, But……

Prime = {1, 2, 3, 5 ,7, 9, 11, 13, 15, 17, 19, …. }
Quanratic time complexity 문제라도 n 이 워낙 큰값이면 무지막지한 시간)

RSA is safe (for now)

prime number n 으로 부터 p, q 를 알아내는 일!

- Suppose someone can factor my 5-digit *n* in 1 ms,
- At this rate, to factor a 10-digit number would take 2 minutes.
- … to factor a 15-digit number would take 4 months.
- … 20-digit number … 30,000 years.
- … 25-digit number… 3 billion years.
- We're safe with RSA! (at least, from factoring with digital computers)

Certificate Authorities

- **How do we know we have the right public key for someone?**
- *Certificate Authorities* sign digital certificates indicating authenticity of a sender who they have checked out in the real world.
- Senders provide copies of their certificates along with their message or software.
- But can we trust the certificate authorities? (only some)

Encryption Support in Databases

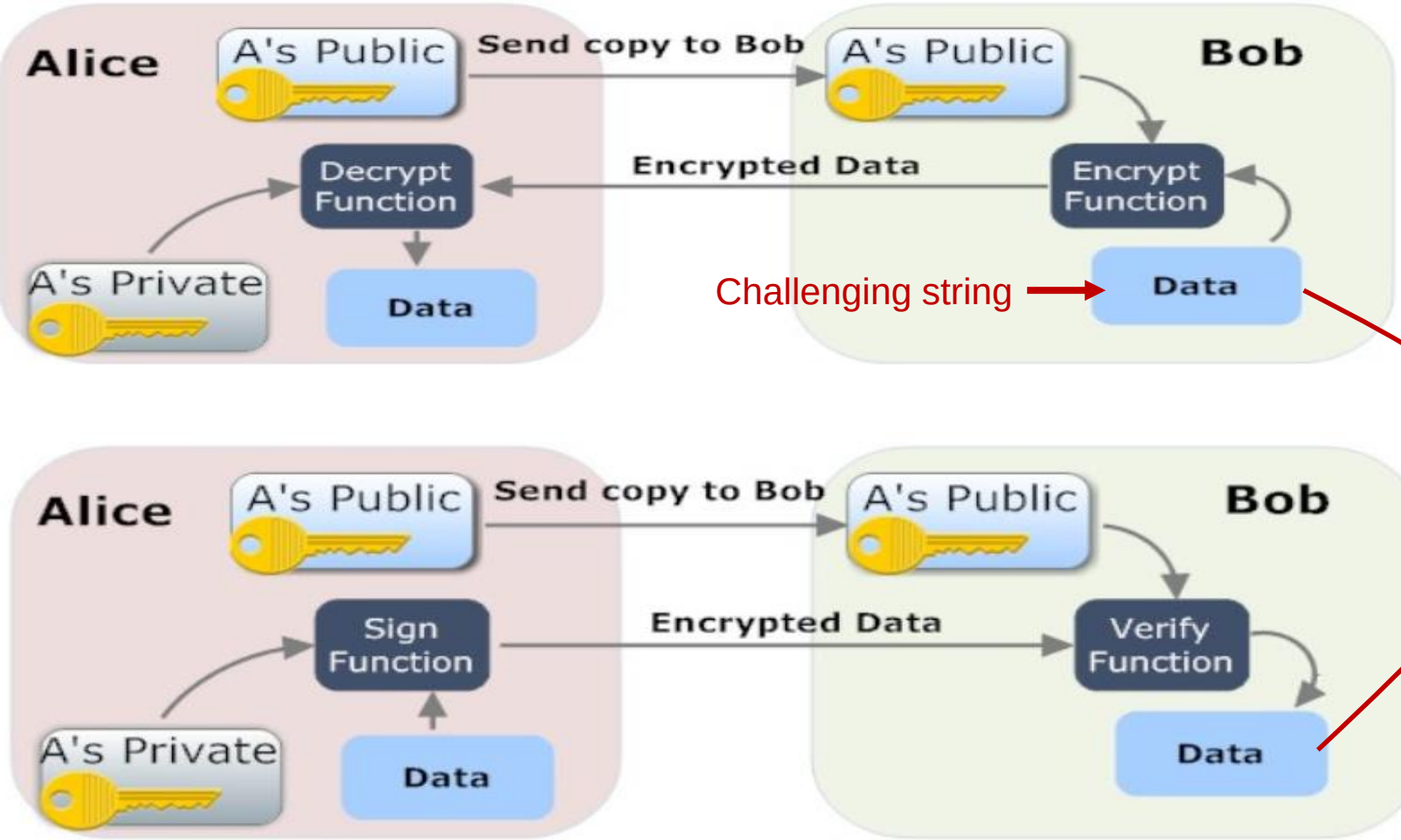
- Different levels of encryption support in database:
  - **Disk-block-level encryption**
    - every disk block encrypted using key available in DBMS
    - Even if attacker gets access to database data, decryption cannot be done without access to the key
  - **Entire relations, or specific attributes of relations**
    - Only sensitive parts of relations (such as credit card number) are encrypted
    - However, attributes involved in primary/foreign key constraints cannot be encrypted (because indexing is not allowed on encrypted attributes)
- Storage of encryption or decryption keys
  - A single master encryption key used to protect multiple encryption/decryption keys stored in database
- Alternative: encryption/decryption is done in application, before sending values to the database

Authentication using Encryption

- **Password based authentication** is widely used, but is susceptible to sniffing on a network
  - Sniff: 킁킁거리며 냄새맡다
  - Eavesdrop: 도청하다
  - Masquerade: 가면을 쓰고 행동하다
- **Challenge-response systems** avoid transmission of passwords
  - DB sends a (randomly generated) challenge string to user
  - User encrypts string and returns result
  - DB verifies identity by decrypting result
  - Can use public-key encryption system
    - DB sending a message encrypted using user's public key
    - User decrypting and sending the message back

- 참고: 자동가입 방지 문자 시스템
  - 생성된 숫자 혹은 모양을 사용자보고 직접 입력하도록 요청
  - Spam 성향 SW 가 site에 회원가입하는것을 방지

Challenge-Response System



일치하면
Alice는 진짜
Alice라고 인증

Digital Signatures

Repudiate: 부인하다

- Public-Key encryption mechanism authenticates the data

- 보안서류전달: A가 B에게 안전하게 서류 D를 보낼때는 B의 public key로 encrypt해서 send, B는 private key로 D를 decrypt 해서 내용을 볼수있다

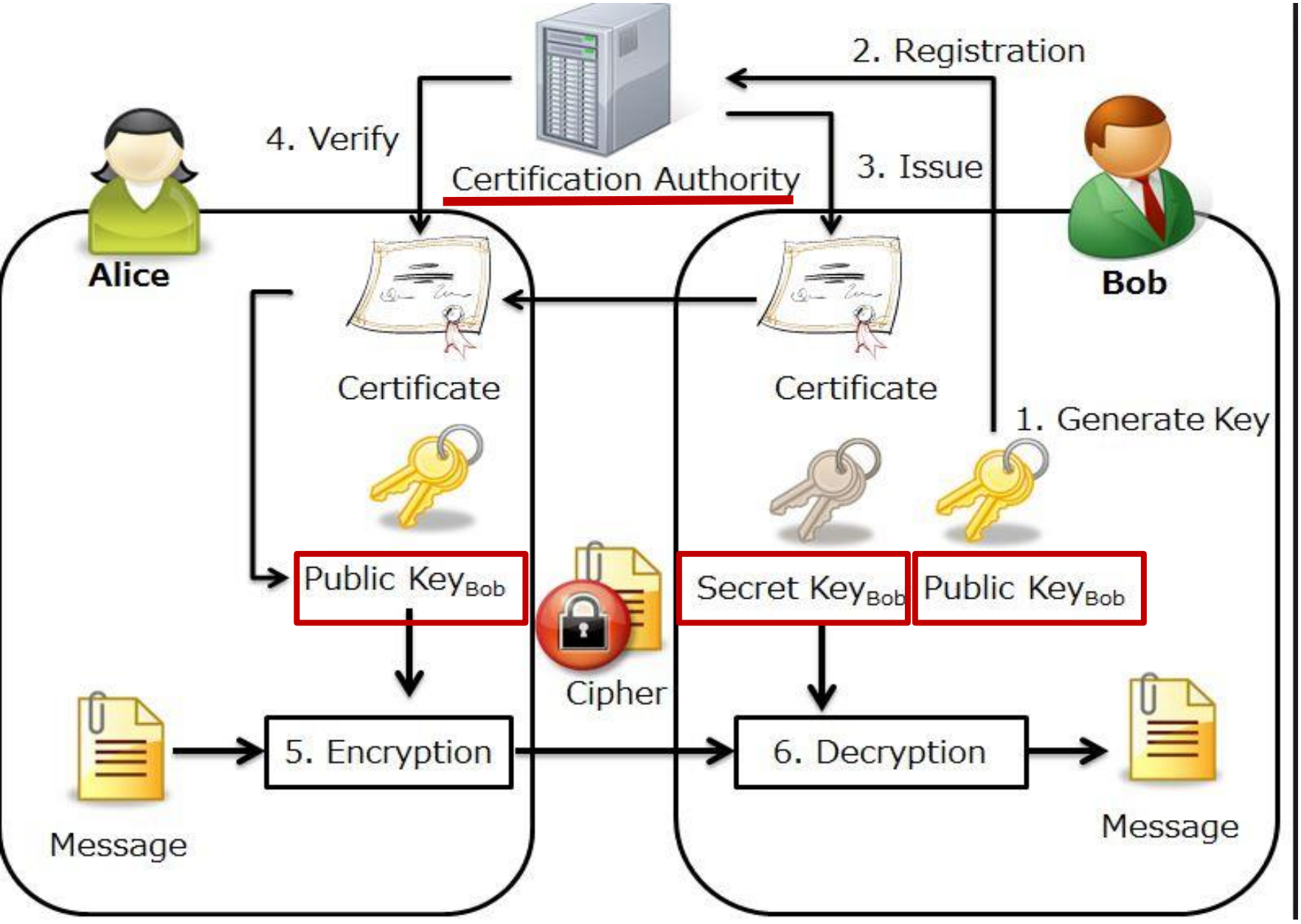
- **A의 "진짜" 서명이 들어있는 서류를 받아야 하는 상황**
- User A use A's private key to encrypt data, and B can verify authenticity by using A's public key to decrypt data
  - Only holder of private key could have created the encrypted data

- Digital signatures help ensure **nonrepudiation (부인방지)**
  - Sender cannot later claim to have not created the data

Digital Certificates [1/2]

- **Digital certificates** are used to verify authenticity of public keys
- Problem: when you communicate with a web site, how do you know if you are talking with the genuine web site or an imposter (사칭하는 사람)?
  - Solution: use the public key of the web site
  - Problem: how to verify if the public key itself is genuine?
- Solution:
  - Every client (e.g., browser) has public keys of a few root-level **certification authorities**
  - A site can get its name/URL and public key signed by a certification authority
    - signed document is called a **certificate**
  - Client can use public key of certification authority (인증기관) to verify certificate
  - Certification authority (CA)
    - presents its own public-key certificate signed by a higher level authority
    - uses its private key to sign the certificate of other web sites/authorities
  - Multiple levels of certification authorities can exist

Digital Certificate Mechanism



Digital Certificates [2/2]

- Authenticating users using digital certificates
- The user must submit a digital certificate containing her pubic key to a site
  - It verifies that the certificate has been signed by a trusted authority
- The user's public key then can used in a challenge-response system to ensure that the user possesses private key
- Thereby authenticating the user