Universitetet i Agder
Fakultet for økonomi- og samfunnsfag

EKSAMEN

Emnekode:	**IS-202**
Emnenavn:	**Programming related topics**
Dato:	**14 December 2007**
Varighet:	**0900-1300**
Antall sider inkl. forside:	6
Målform:	**English**
Tillatte hjelpemidler:	**Any printed or handwritten material**
Merknader:	

Question 1

Explain what version control systems (e.g. subversion) can do, and why they are used in system development.
(Counts 30%)

Question 2

Almost all web applications have a significant amount of static content (e.g. plain html files, images and other data files). The static content is usually deployed as ordinary files inside the "web" folder in the web application archive (the war-file).
When a browser sends a request for static content, an image for example, the server must convert the URL to a filename, read the file and send its contents over the network to the browser. If this is impossible for any reason, e.g. the file does not exist; the server must send an appropriate error message to the browser.
In Tomcat this functionality is provided by the default servlet, which is invoked when a browser sends a request for a URL that does not match any other servlet. Appendix 1 contains the source code for a very simple servlet that handles static content.

a) The FileServlet is far from perfect. For example it will not handle images (why?). Write a list of possible improvements, with a short description of what to do, and why for each item.
(Counts 20%)

An alternative to storing the static content in the file system is to use a database. Instead of storing data in a file, the data is stored in a database table as a BLOB (binary large object), along with the file metadata (file name, file size etc.). This is particularly attractive in applications where content may be added (or changed) over time. Some examples are wikis, the course catalog of the university and the product catalog in a webshop.

b) Rewrite the FileServlet to get static content from a database rather than the file system. Instead of using the URL to create a File object, the servlet will have to search for the filename in the database, and read from a Blob instead of a file when sending the content to the client.
The following information is provided:
   - Table definition and relevant query (appendix 2)
   - Excerpts from ResultSet javadoc (appendix 3)
   - Excerpts from Blob javadoc (appendix 4)

   (Counts 50%)

Appendix 1 - FileServlet.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * This servlet serves static content.
 *
 * @author Even Aaby Larsen
 */
public class FileServlet extends HttpServlet {


    /** The handling steps */
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        handleRequest(req, resp);
    }


    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        handleRequest(req, resp);
    }


    protected void handleRequest(HttpServletRequest req,
                                 HttpServletResponse resp)
    throws ServletException, IOException {
        printHeaders(req, resp);
        serveFile(req, resp);

    }


    private void printHeaders(HttpServletRequest req,
                              HttpServletResponse resp) {
        resp.setContentType("text/html");
        resp.setHeader("Cache-control", "no-cache");
        resp.setHeader("Expires", "0");
    }


    private void serveFile(HttpServletRequest req,
                           HttpServletResponse resp)
            throws IOException {
        ServletOutputStream out = resp.getOutputStream();
        File filename = new File("./webapps"+req.getRequestURI());
        FileInputStream in = new FileInputStream(filename);

        byte[] buf = new byte[1024];
        for (int n = in.read(buf); n > -1; n = in.read(buf))
            out.write(buf, 0, n);
    }
}
```

Appendix 2 – Table static_content

Table definition:

```
CREATE TABLE static_content (
  filename VARCHAR2(128) NOT NULL,
  content_type VARCHAR2(32),
  filesize NUMBER,
  file BLOB,

  CONSTRAINT static_content_pk PRIMARY KEY (filename)
);
```

Filename is the name of the "file" and the primary key.
Content_type is the mime type of the contents of the blob (e.g. text/html for a html file or image/jpeg for a jpeg image).
Filesize is the size of the blob in bytes.
File is the actual data.

You can assume that a DataSource named "jdbc/contendDB" has been defined, and that the static_content table can be accessed using this DataSource.

The following query can be used in a PreparedStatement to retrieve a file from the table by name:

SELECT filename, content_type, filesize, file FROM static_content WHERE filename like ?

Appendix 3 – Interface ResultSet

java.sql

Interface ResultSet

A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

String getString(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as a String in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: the column value; if the value is SQL NULL, the value returned is null
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

int getInt(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as an int in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: the column value; if the value is SQL NULL, the value returned is 0
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

Blob getBlob(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as a Blob object in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: a Blob object representing the SQL BLOB value in the specified column
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

Appendix 4 – Interface Blob

The representation (mapping) in the Java programming language of an SQL BLOB value. An SQL BLOB is a built-in type that stores a Binary Large Object as a column value in a row of a database table.
The Blob interface provides methods for getting the length of an SQL BLOB (Binary Large Object) value, for materializing a BLOB value on the client, and for determining the position of a pattern of bytes within a BLOB value. In addition, this interface has methods for updating a BLOB value.

long length()
throws SQLException
Returns the number of bytes in the BLOB value designated by this Blob object.
Returns: length of the BLOB in bytes
Throws:
SQLException - if there is an error accessing the length of the BLOB

InputStream getBinaryStream()
throws SQLException
Retrieves the BLOB value designated by this Blob instance as a stream.
Returns: a stream containing the BLOB data
Throws:
SQLException - if there is an error accessing the BLOB value

Universitetet i Agder
Fakultet for økonomi- og samfunnsfag

EKSAMEN

Emnekode:	**IS-202**
Emnenavn:	**Programmeringsrelaterte emner**
Dato:	**14. desember 2007**
Varighet:	**0900-1300**
Antall sider inkl. forside:	6
Målform:	**Norsk**
Tillatte hjelpemidler:	**Alle trykte og skrevne**
Merknader:	

Oppgave 1

Forklar hva et versjonskontrollsystem (som f.eks. subversjon) kan gjøre, og hvorfor de blir brukt i systemutvikling.
(Teller 30%)

Oppgave 2

Nesten alle web-applikasjoner har mye statisk innhold (vanlige html filer, bilder og andre data filer). Det statiske innholdet blir normalt lagt i "web"-mappen i war-filen.
Når en nettleser sender et request etter statisk innhold, f.eks. et bilde, må serveren "oversette" URL til et filnavn med bane, lese filen og sende innholdet over nettet til nettleseren. Hvis dette av en eller annen grunn ikke er mulig, f.eks. hvis filen ikke finnes, sender serveren en feilmelding.
I Tomcat er denne funksjonaliteten implementert i default-servleten. Den blir kalt når det ikke finnes noen annen servlet som matcher URL i requestet. Du finner en meget forenklet utgave av en servlet som sender statiske filer i Vedlegg 1.

a) Klassen FileServlet er på ingen måte perfekt. Den kan for eksempel ikke håndtere bilder (hvorfor ikke?). Lag en liste over mulige forbedringer, med en kort beskrivelse av hva som må gjøres og hvorfor, for hvert punkt på listen.
(Teller 20%)

I stedet for å lagre statisk innhold i filsystemet kan vi bruke en database. Da lagres innholdet som BLOBer (binary large object), sammen med opplysninger om filen (filnavn, størrelse osv.). Dette er spesielt interessent i applikasjoner hvor statisk innhold blir lagt til (og evt. endret) over tid. Eksempler på dette er Wiki'er, studiehåndboken og varekatalogen i nettbutikker.

b) Skriv om FileServlet slik at den henter statisk innhold fra databasen i stedet for filsystemet. I stedet for å bruke URLen til å lage et File-objekt, må servlet slå opp i databasen på filnavnet, og lese fra en Blob i stedet for fil når den sender innholdet til nettleseren.
Følgende informasjon er vedlagt:
   - Tabelldefinisjon og sql spørrring (Vedlegg 2)
   - Utdrag fra ResultSet javadoc (Vedlegg 3)
   - Utdrag fra Blob javadoc (Vedlegg 4)

   (Teller 50%)

Appendix 1 - FileServlet.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * This servlet serves static content.
 *
 * @author Even Aaby Larsen
 */
public class FileServlet extends HttpServlet {


    /** The handling steps */
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        handleRequest(req, resp);
    }


    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        handleRequest(req, resp);
    }


    protected void handleRequest(HttpServletRequest req,
                                 HttpServletResponse resp)
    throws ServletException, IOException {
        printHeaders(req, resp);
        serveFile(req, resp);

    }


    private void printHeaders(HttpServletRequest req,
                              HttpServletResponse resp) {
        resp.setContentType("text/html");
        resp.setHeader("Cache-control", "no-cache");
        resp.setHeader("Expires", "0");
    }


    private void serveFile(HttpServletRequest req,
                           HttpServletResponse resp)
            throws IOException {
        ServletOutputStream out = resp.getOutputStream();
        File filename = new File("./webapps"+req.getRequestURI());
        FileInputStream in = new FileInputStream(filename);

        byte[] buf = new byte[1024];
        for (int n = in.read(buf); n > -1; n = in.read(buf))
            out.write(buf, 0, n);
    }
}
```

Appendix 2 – Table static_content

Table definition:

```
CREATE TABLE static_content (
  filename VARCHAR2(128) NOT NULL,
  content_type VARCHAR2(32),
  filesize NUMBER,
  file BLOB,

  CONSTRAINT static_content_pk PRIMARY KEY (filename)
);
```

Filename is the name of the "file" and the primary key.
Content_type is the mime type of the contents of the blob (e.g. text/html for a html file or image/jpeg for a jpeg image).
Filesize is the size of the blob in bytes.
File is the actual data.

You can assume that a DataSource named "jdbc/contendDB" has been defined, and that the static_content table can be accessed using this DataSource.

The following query can be used in a PreparedStatement to retrieve a file from the table by name:

SELECT filename, content_type, filesize, file FROM static_content WHERE filename like ?

Appendix 3 – Interface ResultSet

java.sql

Interface ResultSet

A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

String getString(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as a String in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: the column value; if the value is SQL NULL, the value returned is null
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

int getInt(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as an int in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: the column value; if the value is SQL NULL, the value returned is 0
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

Blob getBlob(String columnLabel)
throws SQLException
Retrieves the value of the designated column in the current row of this ResultSet object as a Blob object in the Java programming language.
Parameters:
columnLabel - the label for the column specified with the SQL AS clause. If the SQL AS clause was not specified, then the label is the name of the column
Returns: a Blob object representing the SQL BLOB value in the specified column
Throws:
SQLException - if the columnLabel is not valid; if a database access error occurs or this method is called on a closed result set

Appendix 4 – Interface Blob

The representation (mapping) in the Java programming language of an SQL BLOB value. An SQL BLOB is a built-in type that stores a Binary Large Object as a column value in a row of a database table.
The Blob interface provides methods for getting the length of an SQL BLOB (Binary Large Object) value, for materializing a BLOB value on the client, and for determining the position of a pattern of bytes within a BLOB value. In addition, this interface has methods for updating a BLOB value.

long length()
throws SQLException
Returns the number of bytes in the BLOB value designated by this Blob object.
Returns: length of the BLOB in bytes
Throws:
SQLException - if there is an error accessing the length of the BLOB

InputStream getBinaryStream()
throws SQLException
Retrieves the BLOB value designated by this Blob instance as a stream.
Returns: a stream containing the BLOB data
Throws:
SQLException - if there is an error accessing the BLOB value