CS 580 Client-Server Programming
Fall Semester, 2000
Doc 14 Html Forms & Assignment 1 Comments
Contents

References

Past 580 lectures

Student papers

HTML Forms
GET

Sample Form

```
<FORM ACTION="http://www-rohan.sdsu.edu/faculty/whitney/foobar"
METHOD="GET">

  <INPUT TYPE="checkbox" NAME="top" VALUE="on">
    Press me.<P>

  <INPUT TYPE="checkbox" NAME="bottom" VALUE="on">
    Press me.<P>

  <INPUT TYPE="text" NAME="userText" SIZE= 30>  Type here<P>

  <INPUT TYPE="radio" NAME="dial" VALUE="low">
    Press me.<P>

  <INPUT TYPE="radio" NAME="dial" VALUE="high">
    Press me.<P>

  <INPUT TYPE="submit" VALUE="Submit"> <P>
</FORM>
```

Resulting Request

```
GET
/faculty/whitney/foobar/CGIEcho.cgi?top=on&bottom=on&userText=n
m&dial=low HTTP/1.0
```

Form Data

Each form item has a name and a value

Text box value is the text typed by the user

If a form has items with values they are sent as:

name1=value1&name2=value2 etc

Button values sent only if button is selected

Text value sent only if user provides text

x-www-form-urlencoded Format

Form data is sent in x-www-form-urlencoded format

- 'a' - 'z', 'A' - 'Z', and '0' - '9' remain the same.
- Space character ' ' is converted into a plus sign '+'.
- All other characters are converted into the 3-character string

  "%xy", where xy is the two-digit hexadecimal representation of the lower 8-bits of the character.

Java Encoding/Decoding Classes

Java provides classes to encode/decode x-www-form-urlencoded format

java.net.URLEncoder

```
public static String encode(String s)
```

java.net.URLDecoder

```
public static String decode(String s) throws Exception
```

POST

Post is the same as GET except the form data in the body of the http request

```
<FORM ACTION="http://www-
rohan.sdsu.edu/faculty/whitney/fooBar"
METHOD="POST">
   <INPUT TYPE="checkbox" NAME="top" VALUE="on">
     Press me.<P>
   <INPUT TYPE="checkbox" NAME="bottom" VALUE="on">
     Press me.<P>
   <INPUT TYPE="text" NAME="userText" SIZE= 30>  Type here<P>
   <INPUT TYPE="radio" NAME="dial" VALUE="low">
     Press me.<P>
   <INPUT TYPE="radio" NAME="dial" VALUE="high">
     Press me.<P>
   <INPUT TYPE="submit" VALUE="Submit"> <P>
</FORM>
```

The request:

```
POST /faculty/whitney/fooBar HTTP/1.0
```

Request body

```
bottom=on&userText=hi&dial=high
```

Form Examples

For more examples see:

http://www.eli.sdsu.edu/courses/spring97/cs596/notes/cgiExamplesRohan/cgiExamplesRohan.html

Creating HTML in Java
First Method - By Hand

```
StringBuffer page = new StringBuffer();
page.append( "<HTML><HEAD><TITLE>" );
page.append( pageTitle);
page.append( "</TITLE></HEAD><BODY>");
```

etc.

The gets old fast.

Try creating a table this way!

Creating HTML in Java
Second Method - Don't
Use sdsu.html.Template

Basic idea:

- Create html templates with variables
- Store the temple in a file
- At run time read template and replace variable with actual values

The methods

public static Template fromFile(String fileName)

public Template(String templateText)

public void replace(String variableName, HTML variableValue)

public void replace(String variableName, String variableValue)

public void setVariableMarker(String newMarker)

public String toString()

Example

Create a file with HTML: say payUp

```
<HTML>
<HEAD>
<TITLE>Collection Agency</TITLE>
</HEAD>
<BODY>
<H2>
<center>Time To Pay Up</center></H2>
<P>
Dear @@@deadBeat@@@;
<P>According to our records you owe us @@@amount@@@.
Pay now or <STRONG>else</STRONG>!

Sincerely
  @@@me@@
```

Program

```
class ShowTemplate
  {
  public  static  void  main( String  args[] ) throws Exception
    {
    Template overDue = Template.fromFile( "payUp" ) ;
    overDue.replace( "deadBeat", "George" );
    overDue.replace( "amount", "$1,000,000" );
    overDue.replace( "me", "Nice Guy" );
    System.out.println( overDue );
    }
  }
```

Output

<HTML>
<HEAD>
<TITLE>Collection Agency</TITLE>
</HEAD>
<BODY>
<H2>
Time To Pay Up </H2>
<P>
Dear George;
<P>According to our records you owe us $1,000,000.
Pay now or <STRONG>else</STRONG>!

Sincerely
Nice Guy

Creating HTML in Java
Third Method - Use sdsu.html

Create classes to represent the elements in an html page

Button	Image
Form	SelectionList
Formatter	TextInput
HTMLTable	WebPage

WebPage

Represents a page

Methods:

WebPage()
WebPage(String pageTitle)

append(Object)
appendLineBreak()

clearPageBody()

getPageBody()
getPageEnding()
getPageHeader()
Returns in HTML format the header information of the page.

setBackgroundColor(String color)
setBackgroundImage(String imageUrl)
setTextColor(String color)

toString()

Example

```
class SimplePage
  {
  public  static  void  main( String  args[] )
    {
    WebPage simple = new WebPage( "Example" );
    simple.setTextColor( WebPage.WHITE);
    simple.setBackgroundColor( WebPage.BLACK);
    simple.append( "Hi Mom");

    System.out.println( simple );
    }
  }
```

HTML Output

```
<!DOCTYPE HTML SYSTEM "html.dtd">
<!-- Document generated via code written using java library sdsu.html
version 0.8>
<HTML>
<HEAD>
<TITLE>Example
</TITLE>
</HEAD>
<BODY TEXT="#FFFFFF" BGCOLOR="#000000">Hi Mom
</BODY>
</HTML>
```

Button

All buttons must be in a form

Some static button methods for creating button types

checkBox(String name, String value)
  Has name and value)
  If checked returns name=value

hidden(String name, String value)
  Store name-value pair hidden from users view

radio(String name, String value)
  Has name and value
  All radio buttons with same name are in same group
  Selected button returns name=value

reset(String value)
  Returns form to original state when pressed

submit(String value)
  Sends form values to form URL

SelectionList

Create a dropdown menu or a scrollable list in an html form

scrolling(String name, int size)
  Shows items in a scrolling window
  Has a name

```
int windowSize = 4;
String name = "Fun";
SelectionList demo = SelectionList.scrolling(name,windowSize);
```

DropDownMenu(String name)
  Shows items in a drop down menu
  Has a name

```
String name = "drop";
SelectionList demo = SelectionList.dropDownMenu(name);
```

Adding items to list

```
String itemNameAndReturnValue = "Hi";
String itemName = "Bye";
String itemReturnValue = "a";

demo.append( itemNameAndReturnValue );
demo.append( itemName, itemReturnValue );
```

List name and return value of selected item is returned with form

TextInput

Must be in a form

name is set to CGI program, its value is the text in the input area

password(name, numberOfColumns)
Create a text input area for entering passwords.

scrolling(name, numberOfColumns, numberOfRows)
Create a scrolling text input area.

singleLine(name, numberOfColumns)
Create a single-line text input area.

singleLine(name, numberOfColumns, maxNumberOfCharacters)
Create a single-line text input area.

setInitalText(String)
Sets the initial text seen in the text input.

Example

```
TextInput sample = TextInput.passwird("userPassword", 5);
```

Form

A container for Buttons, SelectionLists, TextInputs, and any other HTML items/tags you wish to add.

Needs at least one submit button.

Creation

```
String url = "http://www.eli.sdsu.edu/cgiExamples/test.cgi";

Form get = Form.methodGet( url );
Form post = Form.methodPost( url );
```

Methods

```
append( ... );
appendLine( ... );
```

Example

```
Form simple = Form.methodGet("http://www.eli.sdsu.edu/cgi
bin/cgiExamples/CGIEcho");

// Show some Button examples
simple.appendLine( "Programmer Desire" );
simple.appendLine( Button.radio( "desire", "low"), "Low");
simple.appendLine( Button.radio( "desire", "med"), "Meduim");
simple.appendLine( Button.radio( "desire", "high"), "High");
simple.append( Button.hidden( "suprise", "can't see me"));
simple.appendLine();

// Show some TextInput examples
simple.appendLine( "Please type in your name", TextInput.singleLine(
"name", 8 ));

// Show some SelectionList examples
SelectionList language =  SelectionList.scrolling( "language", 4);
language.append( "C++", "a" );
language.append( "C");
language.append( "Java");
language.append( "Ada" );
language.append( "Pascal" );

simple.append( Button.reset( "Clear Form"));
simple.append( Button.submit( "Send Now"));

WebPage test = new WebPage( "Form Test");
test.append( simple );
```

HTMLTable

HTMLTable(int, int)
Create a new table with given number of rows and columns.
HTMLTable(Table)
Create a new html table with element given by the sdsu.util.Table

alignCenter()
alignLeft()
alignRight()

elementAt(int, int)

makeColumnAHeader(int)
makeRowAHeader(int)

setBorderWidth(int)
setCaption(Formatter)
Sets text caption displayed above the table.

setElementAt(HTML, int, int)
setElementAt(Object, int, int)

setWidth(int)
Suggests a width in pixels of the table.

setWidthPercent(int)
Suggests a width for the table in percent of window width.

toString()

Formatter

Formats text and other items

alignCenter() alignLeft() alignRight()

append(HTML)
Append the existing HTML object to the current end of the text.

append(String)
appendBold(String)
Append the string to the current end of the text as bold text

appendHeading(String, int)
appendHTMLTags(String)
Appends the string without escaping special characters.
appendItalic(String)
appendLineBreak()
appendLink(String, url)
Append the string to the current end of the text as linked text.
appendMailLink(String)
Appends a mailTo reference.

makeBlockQuote()
makeParagraph()
makePreformatted()

toString()
Converts the text to an string with proper html tags

Simple Concurrency

```
public static void main(String[] args) throws IOException {
   ServerSocket acceptor = new ServerSocket(0);
   System.out.println("On port " + acceptor.getLocalPort());

   while (true) {
      Socket client = acceptor.accept();
      ServerThread request = new ServerThread( client);
      Request.run();
   }
}
```

Comments on Homework 1
Code Layout

Alignment is important

```
public abstract class HttpTransaction
 {
 HashMap headers = new HashMap();
 byte[] body;

    abstract void  parseFirstLine(ByteReader input );
 abstract String firstLineToString();
  public void fromStream( InputStream input) throws IOException
 {
    ByteReader bufferedInput = new ByteReader( input);

     parseFirstLine(bufferedInput);
      if ( hasHeaders() )
                        parseHeaders(bufferedInput);
    if (hasBody() )
        parseBody(bufferedInput);
     }

    public byte[] toBytes()
   {
  if (!hasBody())
      return toString().getBytes();
     String headerString = toString();
    int length = headerString.length() + body.length;
     byte[]  transaction = new byte[length];
    System.arraycopy(headerString.getBytes(), 0, transaction, 0, headerString.length());
       System.arraycopy(body, 0, transaction, headerString.length(), body.length);
   return transaction;
        }

    public boolean hasContentLength()
        {
        return (headers.containsKey( "Content-length" ) ||
            headers.containsKey( "Content-Length" ));
        }
```

Comment Alignment

```
public abstract class HttpTransaction
    {
    HashMap headers = new HashMap();
    byte[] body;

    abstract void  parseFirstLine(ByteReader input );
    abstract String firstLineToString();
    public void fromStream( InputStream input) throws IOException
        {
//Create a byte reader
        ByteReader bufferedInput = new ByteReader( input);
//Parse a line
        parseFirstLine(bufferedInput);
//Parse headers if exists
        if ( hasHeaders() )
            parseHeaders(bufferedInput);
//Parse body is exisits
        if (hasBody() )
            parseBody(bufferedInput);
        }
/*-------------------------
* Method here
*-------------------------*/
    public byte[] toBytes()
        {
//See if there is a body
        if (!hasBody())
            return toString().getBytes();
//Get headers
        String headerString = toString();
        int length = headerString.length() + body.length;
        byte[]  transaction = new byte[length];
        System.arraycopy(headerString.getBytes(), 0, transaction, 0, headerString.length());
        System.arraycopy(body, 0, transaction, headerString.length(), body.length);
        return transaction;
        }

    public boolean hasContentLength()
        {
// return true if has content length
        return (headers.containsKey( "Content-length" ) ||
            headers.containsKey( "Content-Length" ));
        }
```

LineWrap

(May not show in html)

```
	public byte[] toBytes()
		{
		if (!hasBody())
			return toString().getBytes();
		String headerString = toString();
		int length = headerString.length() + body.length;
		byte[]  transaction = new byte[length];
		System.arraycopy(headerString.getBytes(), 0,
transaction, 0, headerString.length());
		System.arraycopy(body, 0, transaction,
headerString.length(), body.length);
		return transaction;
		}
			{
			String requestLine = readLine(input);
			while (requestLine.length() > 0 )
				{
				int separatorIndex = requestLine.indexOf(':');
				String name = requestLine.substring(0,
separatorIndex);
				String value =
requestLine.substring(separatorIndex + 1).trim();
				headers.put( name, value);
				requestLine = readLine(input);
				}
			}
		catch (IOException readError )
```

How to Avoid Line wrap

- Create a file that contains:

123456789112345678921234567893123456789412345678 95 etc

- Print the file using the printer/computer you are going to use to print your assignments

- See how many characters you can get in one line

- Add the following comment to each file

//2345678911234556 78 etc to your page limit

Names

Use the Java standard

thisIsTheJavaStandardNameStyleForMethodNamesAndVariables

ClassName

Comments

Don’t waste your time and the reader's time with useless comments

```
public class Proxy {

   //Constructor
   public Proxy() { blah }

   //main method
   public static void public static void main( String[] args ) {
      //Create a new socket
      ServerSocket acceptor = new ServerSocket(0);
      //Print out the port number
      System.out.println("On port " + acceptor.getLocalPort());

      //runs forever, until program is manually killed
      while (true) {
         //Accept a new connection
         Socket client = acceptor.accept();
         //Process the client request
         processRequest(
            client.getInputStream(),
            client.getOutputStream());
         client.close();
      }
   }
}
```

Socket IO is not like File IO

Data on a socket is delivered in packets

One packet may not contain all the data

Some packets may be delayed

```
InputStream fromServer = getServerStream();

byte[] headers = new byte[10000];

//read headers
int charsRead = fromServer.read( headers );

//now parse headers
```

IO needs to be reasonable fast

The following is not reasonably fast

```
InputStream fromServer = getServerStream();

StringBuffer theData = new StringBuffer();
int input;
while (-1 != (input= fromServer.read() )
   {
   theData.append( (char) input);
   if ( -1 < theData.toString.indexOf( "\r\n\r\n" ) )
      return theData.toString();
   }
```

Avoid using fields as method parameters

```
public class Proxy {
   int port;
   String host;
   String firstLine;
   InputStream fromBrowser;
   Socket toServer;

   public main() {
      blah;
      getFirstLine();
      getPortAndHost();
      connectToServer();
      blah;
   }

   private getFirstLine() {
      firstLine = fromBrowser.readLine();
   }

   private getPortAndHost() {
      do stuff with firstLine
      host = blah;
      port = more blah;
   }

   private connectToServer() {
      toServer = new Socket( host, port);
   }
```

```
public class Proxy {

   public main() {
      int port;
      String host;
      String firstLine;
      InputStream fromBrowser;
      Socket toServer;

      blah;
      firstLine = getFirstLine(fromBrowser);
      port = getPort(firstLine );
      host = getHost( firstLine);
      toServer connectToServer(host, port));
   }

   private String getFirstLine(InputStream in) {
      return in.readLine();
   }


   private Socket connectToServer(host, port) {
      return new Socket( host, port);
   }
```