© 2011 Marty Hall

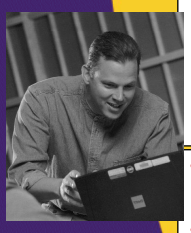
Network Programming: Part I (General Techniques)

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/android-tutorial/

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

© 2011 Marty Hall

For live Android training, please see courses at http://courses.coreservlets.com/.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this Android tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - Android development, JSF 2, servlets/JSP, Ajax, jQuery, Java 6 programming, custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, GWT, RESTful and SOAP-based Web Services

Contact hall@coreservlets.com for details

Topics in This Section

- **Part I (this section): general networking**
  - Socket basics
  - Requesting Internet permission
  - Example: NIST atomic time
  - Aside: simple String formatting and parsing
  - Example: FTP welcome messages
  - Example: validating URLs with HEAD
- **Part II (next section): HTTP-specific approaches**
  - HttpURLConnection
  - HttpClient
  - Examples: Searching Web pages
  - Using JSON
  - Example: remote loan calculations
  - Example: Google translation services

© 2011 Marty Hall

Overview

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Big Idea

- **Many ways to communicate with a server**
  - Socket class
    - Lets you do general-purpose network programming
      - Same as with desktop Java programming
  - HttpURLConnection
    - Simplifies connections to HTTP servers
      - Same as with desktop Java programming
  - HttpClient
    - Simplest way to download entire content of a URL
      - Not standard in Java SE, but standard in Android
  - JSONObject
    - Simplifies creation and parsing of JSON data
      - Not standard in Java SE, but standard in Android

© 2011 Marty Hall

Socket Basics

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Steps for Talking to Server

1. **Create a Socket object**

```
Socket client = new Socket("hostname", portNumber);
```

2. **Create output stream to send data to the Socket**

```
// Last arg of true means autoflush -- flush stream
// when println is called
PrintWriter out =
  new PrintWriter(client.getOutputStream(), true);
```

3. **Create input stream to read response from server**

```
BufferedReader in =
  new BufferedReader
    (new InputStreamReader(client.getInputStream()));
```


Steps for Implementing a Client (Continued)

4. **Do I/O with the input and output Streams**
   - For the output stream, PrintWriter, use print, println, and printf, similar to System.out.print/println/printf
     - The main difference is that you can create PrintWriters for different Unicode characters sets, and you can't with PrintStream (the class of System.out).
   - For input stream, BufferedReader, call read to get a single char or an array of characters, or call readLine to get a whole line
     - Note that readLine returns null if the connection was terminated (i.e. on EOF), but waits otherwise
   - You can use ObjectInputStream and ObjectOutputStream for Java-to-Java communication. Very powerful and simple.
5. **Close the socket when done**

```
client.close();
```

   - Also closes the associated input and output streams

Exceptions

- **UnknownHostException**
  - If host passed to Socket constructor is not known to DNS server.
    - Note that you may use an IP address string for the host
- **IOException**
  - Timeout
  - Connection refused by server
  - Interruption or other unexpected problem
    - Server closing connection does *not* cause an error when reading: null is returned from readLine

Helper Class: SocketUtils

- **Idea**
  - It is common to make BufferedReader and PrintWriter from a Socket, so simplify the syntax slightly
- **Code**

```
public class SocketUtils {

    public static BufferedReader getReader(Socket s) throws IOException {
        return(new BufferedReader
                        (new InputStreamReader(s.getInputStream())));
    }


    public static PrintWriter getWriter(Socket s) throws IOException {
        // Second argument of true means autoflush.
        return (new PrintWriter(s.getOutputStream(), true));
    }
}
```

Requesting Internet Permission

- **Apps that use internet must say so**
  - User will be notified that app wants internet permission, and can deny it. Apps that do not request permission will be denied access by the Android OS
    - It is possible with effort to circumvent this by launching a hidden Web browser that has data embedded in URL
      - See http://dtors.org/2010/08/06/circumventing-android-permissions/
- **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://…" …>
    <uses-sdk android:minSdkVersion="…" />
    <uses-permission android:name="android.permission.INTERNET" />
    …
</manifest>
```


© 2011 Marty Hall

Example: NIST Time Server

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Time from US National Inst. of Standards & Technology (NIST)

- **Idea**
  - NIST sponsors several servers with high-precision time
    - The simplest to read is Daytime Protocol (RFC-867), which returns the time on the second line of the response to a connection. Runs on port 13.
      - See http://www.nist.gov/pml/div688/grp40/its.cfm
      - One popular host name is time-b.nist.gov
- **Approach**
  - Make a Socket connection to time-b.nist.gov on port 13
  - Create a BufferedReader (no PrintWriter needed)
  - Read first line of result and ignore it
  - Read second line of result and print it out

Manifest File (AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.coreservlets.networking"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```

Layout File (res/layout/nist_time.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/..."
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent">
    <Button android:text="@string/nist_button_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="showTime"/>
    <TextView android:id="@+id/time_display"
              android:textSize="@dimen/time_message_size"
              android:layout_width="match_parent"
              android:layout_height="match_parent"/>
</LinearLayout>
```


Values Files

- **res/values/strings.xml**
  - Defines title, prompts, and button labels for all Activities in this section.
    - Used in both portrait and landscape mode.
- **res/values/colors.xml**
  - Defines foreground color for some of the results
    - Used in both portrait and landscape mode.
- **res/values/dimens.xml**
  - Gives font sizes.
    - Used in portrait mode.
- **res/values-land/dimens.xml**
  - Gives font sizes.
    - Used in landscape mode.

Main Activity (NistTimeActivity.java)

```
public class NistTimeActivity extends Activity {
    private String mHost = "time-b.nist.gov";
    private int mPort = 13;
    private TextView mResultDisplay;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.nist_time);
        mResultDisplay =
                (TextView)findViewById(R.id.time_display);
    }
```


Main Activity, Continued (NistTimeActivity.java)

```
    public void showTime(View clickedButton) {
        try {
            Socket socket = new Socket(mHost, mPort);
            BufferedReader in = SocketUtils.getReader(socket);
            in.readLine(); // Ignore leading blank line
            String timeResult = in.readLine();
            mResultDisplay.setText(timeResult);
            socket.close();
        } catch (UnknownHostException uhe) {
            mResultDisplay.setText("Unknown host: " + mHost);
            uhe.printStackTrace(); // View this in DDMS window
        } catch (IOException ioe) {
            mResultDisplay.setText("IOException: " + ioe);
            ioe.printStackTrace(); // View this in DDMS window
        }
    }
}
```

Results

© 2011 Marty Hall

Aside: String Formatting and Parsing

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Formatting and Parsing Strategies

- **Idea**
  - Simple to connect to a server and create Reader/Writer
  - So, hard parts are formatting request and parsing response
- **Approach**
  - Formatting requests
    - Use printf (aka String.format)
  - Parsing response: simplest
    - Use StringTokenizer
  - Parsing response: more powerful
    - Use String.split with regular expressions
  - Parsing response: most powerful
    - Use Pattern and full regex library
      - Not covered in this tutorial

Formatted Output: printf

- **Takes a variable number of arguments**
  - System.out.printf("Formatting String", arg1, arg2, …);
- **Advantages**
  - Lets you insert values into output without much clumsier String concatenation.
  - Lets you control the width of results so things line up
  - Lets you control the number of digits after the decimal point in numbers, for consistent-looking output
- **Very similar to C/C++ printf function**
  - If you know printf in C/C++, you can probably use Java's printf immediately without reading any documentation
    - Although some additions in time formatting and locales
  - Use String.format to get the equivalent of C's sprintf

Simple Example: printf vs. println

- **General idea**
  - Each %s entry in formatting string is replaced by next argument in argument list. %n means newline.
- **Example**

```
public static void printSomeStrings() {
  String firstName = "John";
  String lastName = "Doe";
  int numPets = 7;
  String petType = "chickens";
  System.out.printf("%s %s has %s %s.%n",
                    firstName, lastName, numPets, petType);
  System.out.println(firstName + " " + lastName +
                     " has " + numPets + " " +
                     petType + ".");
}
```

- **Result:**

```
John Doe has 7 chickens.
John Doe has 7 chickens.
```


Controlling Formatting

- **Different flags**
  - %s for strings, %f for floats/doubles, %t for dates, etc.
    - Unlike in C/C++, you can use %s for *any* type (even nums)
- **Various extra entries can be inserted**
  - To control width, number of digits, commas, justification, type of date format, and more
- **Complete details**
  - printf uses mini-language
    - Complete coverage would take an entire lecture
    - However, basic usage is straightforward
  - For complete coverage, see http://download.oracle.com/javase/6/docs/api/java/util/Formatter.html#syntax
- **Most common errors**
  - Using + instead of , between arguments (printf uses varargs)
  - Forgetting to add %n at the end if you want a newline (not automatic)

Printf Formatting Options

	Stands For	Options	Example
%s	**String.** Can output any data type. If arg is Object, toString is called.	**%*width*s** Gives min num of chars. Spaces added to left if needed.	**printf("%8s", "Hi")** outputs **" Hi"**
%d	**Decimal.** Outputs whole number in base 10. Also %x and %o for hex and octal.	**%*width*d %,*width*d** Gives min width; inserts commas.	**printf("%,9d", 1234)** outputs **" 1,234"**
%f	**Floating point.** Lets you line up decimal point and control precision.	**%*width*.*precision*f** **%,*width*.*precision*f** width includes comma and decimal point.	**printf("%6.2f", Math.PI)** outputs **" 3.14"**
%t*x*	**Time (or date).** %tA for day, %tB for month, %tY for year, and many more.	Date now = new Date(); **printf("%tA, %tB ,%tY", now, now, now)** outputs **"Thursday, November 17, 2005"**	
%n	Outputs OS-specific end of line (linefeed on Linux/Unix, CR/LF pair on Windows)		

Printf Example: Controlling Width and Precision

```
public class CEO {
  private String name;
  private double salary; // In billions

  public CEO(String name, double salary) {
    this.name = name;
    this.salary = salary;
  }

  public String getName() { return(name); }

  public double getSalary() { return(salary); }
}
```

Note that you cannot make a class with a "main" method in an Android app and run it from Eclipse in the usual way (R-click, Run As → Java Application). So, the printf and parsing examples are in a separate project called NetworkingSupport. This project also contains the servlet used in the second networking lecture.

Printf Example: Controlling Width and Precision

```
public static void printSomeSalaries() {
  CEO[] softwareCEOs =
    { new CEO("Steve Jobs", 3.1234),
      new CEO("Scott McNealy", 45.5678),
      new CEO("Jeff Bezos", 567.982323),
      new CEO("Larry Ellison", 6789.0),
       new CEO("Bill Gates", 78901234567890.12)};
  System.out.println("SALARIES:");
  for(CEO ceo: softwareCEOs) {
    System.out.printf("%15s: $%,8.2f%n",
            ceo.getName(), ceo.getSalary());
}}
SALARIES:
     Steve Jobs: $    3.12
  Scott McNealy: $   45.57
     Jeff Bezos: $  567.98
  Larry Ellison: $6,789.00
     Bill Gates: $78,901,234,567,890.12
```


Parsing Strings Using StringTokenizer

- **Idea**
  - Build a tokenizer from an initial string
  - Retrieve tokens one at a time with `nextToken`
  - You can also see how many tokens are remaining (`countTokens`) or simply test if the number of tokens remaining is nonzero (`hasMoreTokens`)

```
StringTokenizer tok
  = new StringTokenizer(input, delimiters);
while (tok.hasMoreTokens()) {
  doSomethingWith(tok.nextToken());
}
```

StringTokenizer

- **Constructors**
  - StringTokenizer(String input, String delimiters)
  - StringTokenizer(String input, String delimiters, boolean includeDelimiters)
  - StringTokenizer(String input)
    - Default delimiter set is " `\t\n\r\f`" (whitespace)
- **Methods**
  - nextToken(), nextToken(String delimiters)
  - countTokens()
  - hasMoreTokens()
- **Also see methods in String class**
  - split, substring, indexOf, startsWith, endsWith, compareTo, …
  - Java has good support for regular expressions

Interactive Tokenizer: Example

```
import java.util.StringTokenizer;

public class TokTest {
  public static void main(String[] args) {
    if (args.length == 2) {
      String input = args[0], delimiters = args[1];
      StringTokenizer tok
        = new StringTokenizer(input, delimiters);
      while (tok.hasMoreTokens()) {
        System.out.println(tok.nextToken());
      }
    } else {
      System.out.println
        ("Usage: java TokTest string delimiters");
    }
  }
}
```

Interactive Tokenizer: Result

```
> java TokTest http://www.microsoft.com/~gates/ :/.
http
www
microsoft
com
~gates


> java TokTest "if (tok.hasMoreTokens()) {" "(){. "
if
tok
hasMoreTokens
```


Parsing Strings using the split method of String

- **Basic usage**
  - String[] tokens = mainString.split(delimiterString);
- **Differences from StringTokenizer**
  - *Entire* string is the delimiter (not one-char delimiters)
    - "foobar".split("ob") returns "fo" and "ar"
    - "foobar".split("bo") returns "foobar"
  - You can use regular expressions in the delimiter
    - ^, $, *, +, ., etc for beginning of String, end of String, 0 or more, 1 or more, any one character, etc.
    - See http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#sum
  - Unless you use "+", an empty string is returned between delimiters
    - "foobar".split("o") returns "f", "", and "bar"
    - "foobar".split("o+") returns "f" and "bar"
  - You can supply second argument to split
    - Giving max splits; any extras go in final string

Importance of Regular Expressions

- **Idea**
  - String.split and other methods use regular expressions
  - So do many other languages. Knowing regex syntax is an important part of *every* programmer's repertoire.

From Randall Munroe and xkcd.com

- **Tutorials**
  - http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html#sum
  - http://download.oracle.com/javase/tutorial/essential/regex/

Interactive Tokenizer: Example

```
public class SplitTest {
  public static void main(String[] args) {
    if (args.length == 2) {
      String[] tokens = args[0].split(args[1]);
      for(String token: tokens) {
        if (token.length() != 0) {
          System.out.println(token);
        }
      }
    } else {
      System.out.println
        ("Usage: java SplitTest string delimeters");
    }
  }
}
```

Interactive Tokenizer: Result

```
> java TokTest http://www.microsoft.com/~gates/ :/.
http
www
microsoft
com
~gates

> java SplitTest http://www.microsoft.com/~gates/ :/.
http
www.microsoft.com/~gates/

> java SplitTest http://www.microsoft.com/~gates/ [:/.]+
http
www
microsoft
com
~gates
```


© 2011 Marty Hall

Example: FTP Welcome Messages

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

FTP Server Welcome Messages

- **Idea**
  - When you connect to an FTP server, you usually get a welcome message. The problem is that it can be a variable number of lines long.
    - If multiline, first line should start with "220-" (220 dash) and last line should start with "220 " (220 space)
- **Approach**
  - Connect to server on port 21 and read first line
    - If it does *not* start with "220-", print it and exit
  - If first line *does* start with "220-", read and echo lines until you find one that starts with "220 ".

Example Welcome Messages

- **ftp.microsoft.com**

```
> ftp ftp.microsoft.com
220 Microsoft FTP Service
```

- **ftp.oracle.com**

```
> ftp ftp.oracle.com
220-********************************************************************
220-Oracle FTP Server
220-
220-The use of this FTP server is governed  by United States Export Admini-
220-stration Regulations.  Details of the U.S. Commercial Encryption Export
220-Controls can be  found at the Bureau of Industry and Security web site.
220-All Oracle products are subject to U.S. Export Laws. Diversion contrary
220-to U.S. law is prohibited.
...
220-
220-
   ********************************************************************************

220-
220
```

Example Welcome Message

```
> ftp ftp.ngc.com
220-********************************************************************************
 *                      Access Restricted                                        *

 * The Northrop Grumman computer network is for use, only by authorized    *
 * users, and only for authorized purposes.  Unauthorized sending,         *
 * transmitting, or otherwise disseminating proprietary data or trade      *
 * secrets, private company information or classified data is strictly     *
 * prohibited.  The network and its contents are the exclusive property of *
 * the company.  There is no right of privacy on the part of any
  *
 * individual, regarding any information transmitted, stored or received
  *
 * via the network.  Employees may access the Northrop Grumman Global      *
 * Network only by using tools provided by IT Solutions.  Use of other     *
 * remote access services is strictly prohibited.  Use or access to the
  *
 * network constitutes consent to the company's acceptable use provisions  *
 * contained in Corporate Procedure R1, and to the monitoring, storage,    *
 * retrieval or disclosure or any information transmitted, stored or       *
 * received via the network for any purpose, including employee            *
 * discipline, contractual remedies or criminal prosecutions.              *
 * Passwords must comply with Corporate policy UO J104.                    *
 ********************************************************************************
220 FTP server ready
```


Manifest File (AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.coreservlets.networking"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```

Layout File: Portrait Mode (res/layout/ftp_message.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/..."
              android:orientation="vertical" ...>
    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="wrap_content">
        <TextView android:text="@string/ftp_server_prompt" .../>
        <EditText android:id="@+id/ftp_host" ...
                  android:inputType="textUri">
            <requestFocus></requestFocus>
        </EditText>
    </LinearLayout>
    <Button android:text="@string/ftp_button_text" ...
            android:onClick="showMessage"/>
    <ScrollView android:layout_width="match_parent"
                android:layout_height="match_parent">
        <TextView android:id="@+id/ftp_message_result" .../>
    </ScrollView>
</LinearLayout>
```


Layout File: Landscape Mode (res/layout-land/ftp_message.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/..."
              android:orientation="vertical" ...>
    <TableLayout ... android:stretchColumns="1">
        <TableRow>
            <TextView android:text="@string/ftp_server_prompt" …/>
            <EditText android:id="@+id/ftp_host" ...
                      android:inputType="textUri">
                <requestFocus></requestFocus>
            </EditText>
            <Button android:text="@string/ftp_button_text" ...
                    android:onClick="showMessage"/>
        </TableRow>
    </TableLayout>
    <ScrollView android:layout_width="match_parent"
                android:layout_height="match_parent">
        <TextView android:id="@+id/ftp_message_result" .../>
    </ScrollView>
</LinearLayout>
```

Values Files

- **res/values/strings.xml**
  - Defines title, prompts, and button labels for all Activities in this section.
    - Used in both portrait and landscape mode.
- **res/values/colors.xml**
  - Defines foreground color for some of the results
    - Used in both portrait and landscape mode.
- **res/values/dimens.xml**
  - Gives font sizes.
    - Used in portrait mode.
- **res/values-land/dimens.xml**
  - Gives font sizes.
    - Used in landscape mode.

Main Activity (FtpMessageActivity.java)

```
public class FtpMessageActivity extends Activity {
    private EditText mFtpHost;
    private TextView mFtpMessageResult;
    private static final int FTP_PORT = 21;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ftp_message);
        mFtpHost = (EditText)findViewById(R.id.ftp_host);
        mFtpMessageResult =
          (TextView)findViewById(R.id.ftp_message_result);
    }
```

Main Activity, Continued (FtpMessageActivity.java)

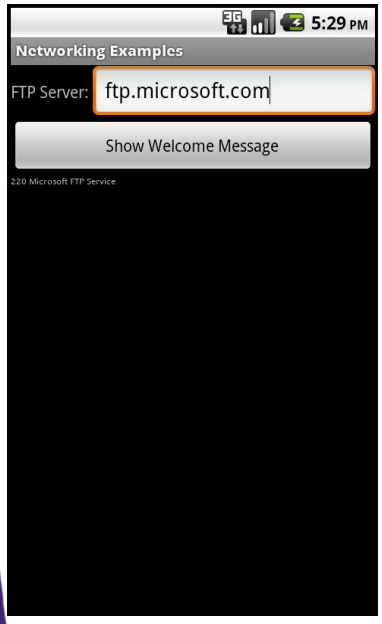
```
public void showMessage(View clickedButton) {
    String host = mFtpHost.getText().toString();
    try {
        Socket socket = new Socket(host, FTP_PORT);
        BufferedReader in = SocketUtils.getReader(socket);
        List<String> results = new ArrayList<String>();
        String line = in.readLine();
        results.add(line);
        if (line.startsWith("220-")) {
            while((line = in.readLine()) != null) {
                results.add(line);
                if ((line.equals("220") ||
                     line.startsWith("220 "))) {
                    break;
                }
            }
        }
        String output = makeOutputString(results);
        mFtpMessageResult.setText(output);
        socket.close();
```

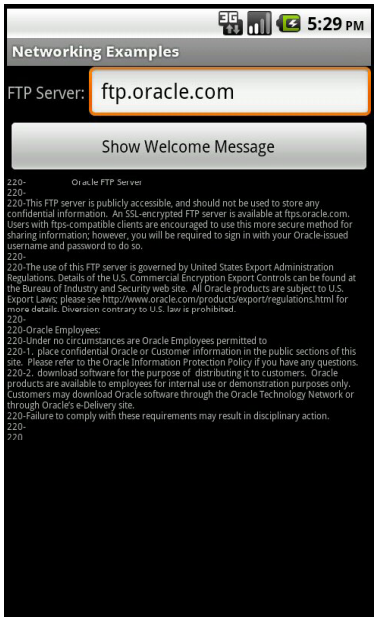

Main Activity, Continued (FtpMessageActivity.java)

```
        } catch (UnknownHostException uhe) {
            mFtpMessageResult.setText("Unknown host: " + host);
            uhe.printStackTrace(); // View this in DDMS window
        } catch (IOException ioe) {
            mFtpMessageResult.setText("IOException: " + ioe);
            ioe.printStackTrace(); // View this in DDMS window
        }
    }

    private String makeOutputString(List<String> results) {
        StringBuilder output = new StringBuilder();
        for (String s: results) {
            output.append(s + "\n");
        }
        return(output.toString());
    }
}
```

Results





© 2011 Marty Hall

Example: Verifying URLs with the HEAD Command

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

The HEAD Command

- **Idea**
  - Web browsers normally send GET or POST requests
    - Server responds with status line, response headers, blank line, and then the document
  - If you only care about status line, send HEAD instead
    - Server responds only with status line and response headers
- **Approach**
  - Parse a URL into the host, port (80 if none) and URI
  - Connect to host on port. Send request for URI plus Host request header (for servers with virtual hosting)
  - Read status line and see if it is 2*xx* (good), 3*xx* (forwarded) or anything else (bad).
    - If forwarded, keep reading to find Location header
- **Note**
  - Here we use no HTTP-specific classes. Next tutorial section covers HttpURLConnection and other HTTP-specific classes.

HTTP Request/Response: GET

- **Request**

GET /foo/bar.html HTTP/1.1
Host: ...
***Header2*: ...**
...
***HeaderN*:**
(Blank Line)

- **Response**

HTTP/1.1 200 OK
Content-Type: text/html
***Header2*: ...**
...
***HeaderN*: ...**
(Blank Line)
<!DOCTYPE ...>
<HTML>
<HEAD>...</HEAD>
<BODY>
...
</BODY></HTML>

HTTP Request/Response: HEAD

- **Request**

HEAD /foo/bar.html HTTP/1.1
Host: ...
***Header2*: ...**
...
***HeaderN*:**
(Blank Line)

- **Response**

HTTP/1.1 200 OK
Content-Type: text/html
***Header2*: ...**
...
***HeaderN*: ...**

Using Telnet to Test Server Commands

- **Telnet**
  - Most people think of telnet as a tool for logging into a remote server on default login port (23)
  - But, it is really more general: a tool for connecting to a remote server on any port and interactively sending commands and looking at results
- **Enabling telnet on Windows 7 or Vista**
  - Starting with Windows Vista, telnet is disabled by default
    - To enable it, see http://technet.microsoft.com/en-us/library/cc771275(WS.10).aspx
    - Or Google for "install telnet windows 7" and above page will come up #1
    - You may also need to turn on local echo – Unix telnet clients are much more convenient

Example: Steve Ballmer's Home Page at Microsoft

```
> telnet www.microsoft.com 80
Trying 207.46.19.254...
Connected to lb1.www.ms.akadns.net.
Escape character is '^]'.
HEAD /presspass/exec/steve/default.aspx HTTP/1.0

HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 182105
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/7.5
X-AspNet-Version: 2.0.50727
VTag: 279105531500000000
P3P: CP="..."
X-Powered-By: ASP.NET
Date: Tue, 02 Aug 2011 17:58:43 GMT
Connection: keep-alive

Connection to lb1.www.ms.akadns.net
closed by foreign host.
```

This URI corresponds to the full URL http://www.microsoft.com/presspass/exec/steve/default.aspx, which is the real bio page for Steve Ballmer.

Also, since microsoft.com is using dedicated hosting, I can use the simpler HTTP 1.0 HEAD request (with no Host header). The Java code will use HTTP 1.1 with the Host header, so that it can also handler sites that use virtual (shared) hosting.

Example: Larry Ellison's Home Page at Microsoft

```
> telnet www.microsoft.com 80
Trying 207.46.19.254...
Connected to lb1.www.ms.akadns.net.
Escape character is '^]'.
HEAD /presspass/exec/larry/default.aspx HTTP/1.0

HTTP/1.1 302 Found
Content-Length: 293
Content-Type: text/html; charset=utf-8
Location:
   http://www.microsoft.com/library/errorpages/smarterror.aspx?as
   pxerrorpath=http%3a%2f%2f207.46.21.164%2fpresspass%2fexec%2fla
   rry%2fdefault.aspx
...
Date: Tue, 02 Aug 2011 18:06:11 GMT
Connection: keep-alive

Connection to lb1.www.ms.akadns.net closed by foreign host.
```

This corresponds to http://www.microsoft.com/presspass/exec/larry/default.aspx, which is a nonexistent page. The page they forward to should return 404 to prevent indexing by search engines.

Manifest File (AndroidManifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.coreservlets.networking"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```


Layout File: Portrait Mode (res/layout/url_checker.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/..."
              android:orientation="vertical" ...>
    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="wrap_content">
        <TextView android:text="@string/url_prompt" .../>
        <EditText android:id="@+id/url_to_test" ...
                  android:inputType="textUri">
            <requestFocus></requestFocus>
        </EditText>
    </LinearLayout>
    <Button android:text="@string/url_checker_button_text" ...
            android:onClick="checkUrl"/>
    <ScrollView android:layout_width="match_parent"
                android:layout_height="match_parent">
        <TextView android:id="@+id/url_validation_result" .../>
    </ScrollView>
</LinearLayout>
```

Layout File: Landscape Mode (res/layout-land/url_checker.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/..."
              android:orientation="vertical" ...>
    <TableLayout ... android:stretchColumns="1">
        <TableRow>
            <TextView android:text="@string/url_prompt" ... />
            <EditText android:id="@+id/url_to_test" ...
                      android:inputType="textUri">
                <requestFocus></requestFocus>
            </EditText>
            <Button android:text="@string/url_checker_button_text"
                    android:onClick="checkUrl"/>
        </TableRow>
    </TableLayout>
    <ScrollView android:layout_width="match_parent"
                android:layout_height="match_parent">
        <TextView android:id="@+id/url_validation_result" .../>
    </ScrollView>
</LinearLayout>
```


Values Files

- **res/values/strings.xml**
  - Defines title, prompts, and button labels for all Activities in this section.
    - Used in both portrait and landscape mode.
- **res/values/colors.xml**
  - Defines foreground color for some of the results
    - Used in both portrait and landscape mode.
- **res/values/dimens.xml**
  - Gives font sizes.
    - Used in portrait mode.
- **res/values-land/dimens.xml**
  - Gives font sizes.
    - Used in landscape mode.

Main Activity (UrlCheckerActivity.java)

```
public class UrlCheckerActivity extends Activity {
    private EditText mUrlToTest;
    private TextView mUrlMessageResult;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.url_checker);
        mUrlToTest =
                (EditText)findViewById(R.id.url_to_test);
        mUrlMessageResult =
                (TextView)findViewById
                        (R.id.url_validation_result);
    }
```


Main Activity, Continued (UrlCheckerActivity.java)

```
    public void checkUrl(View clickedButton) {
        String url = mUrlToTest.getText().toString();
        UrlParser parser = new UrlParser(url);
        String host = parser.getHost();
        int port = parser.getPort();
        String uri = parser.getUri();
        try {
            Socket socket = new Socket(host, port);
            PrintWriter out = SocketUtils.getWriter(socket);
            BufferedReader in = SocketUtils.getReader(socket);
            out.printf("HEAD %s HTTP/1.1\r\n", uri);
            out.printf("Host: %s\r\n", host);
            out.printf("Connection: close\r\n\r\n");
            String serverResult = in.readLine();
            String info = statusInfo(serverResult, in);
            mUrlMessageResult.setText(info);
            socket.close();
```

Main Activity, Continued (UrlCheckerActivity.java)

```
        } catch (UnknownHostException uhe) {
            mUrlMessageResult.setText("Unknown host: " + host);
            uhe.printStackTrace(); // View this in DDMS window
        } catch (IOException ioe) {
            mUrlMessageResult.setText("IOException: " + ioe);
            ioe.printStackTrace(); // View this in DDMS window
        }
    }
```


Main Activity, Continued (UrlCheckerActivity.java)

```
private String statusInfo(String serverResult, BufferedReader in)
        throws IOException {
    StatusLineParser statusLine = new StatusLineParser(serverResult);
    String result;
    if (statusLine.isGood()) {
        result = String.format("Good URL: %s -- %s",
                               statusLine.getStatusCode(),
                               statusLine.getMessage());
    } else if (statusLine.isForwarded()) {
        result = String.format("URL forwarded to %s",
                               location(in));
    } else {
        result = String.format("Bad URL: %s -- %s",
                               statusLine.getStatusCode(),
                               statusLine.getMessage());
    }
    return(result);
}
```

Main Activity, Continued (UrlCheckerActivity.java)

```
private String location(BufferedReader in) throws IOException {
    String line;
    while((line = in.readLine()) != null) {
        if (line.toUpperCase().startsWith("LOCATION")) {
            String[] results = line.split("\\s+", 2);
            return(results[1]);
        }
    }
    return("(Unknown Location)");
}
```


Helper Classes

- **StatusLineParser**
  - Reads status line and breaks it into three parts: HTTP version, status code, and message
    - Also isGood (status code in the 200's), isForwarded (301/302) and isBad (anything else) methods
- **UrlParser**
  - Breaks a URL like http://host:port/path into the host, port, and path parts. Uses 80 if no port specified.
  - Note that the builtin URL class already does this. See next lecture. But Android has no builtin support for protocols other than HTTP, so learning to do it yourself is important.

Results

© 2011 Marty Hall

Wrap-Up

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

More Reading

- **JavaDoc**
  - Socket
    - http://developer.android.com/reference/java/net/Socket.html
  - Pattern (gives regular expression details)
    - http://developer.android.com/reference/java/util/regex/Pattern.html
- **Tutorial: Networking**
  - http://download.oracle.com/javase/tutorial/networking/
    - Not Android specific, but still *very* good networking coverage
- **Chapters**
  - Networking and Web Services
    - From *Android in Action* by Ableson et al
  - Communicating via the Internet
    - From *The Busy Coder's Guide to Android Development* by Mark Murphy (http://commonsware.com/Android/)

Summary

- **Basics**
  - Socket socket = new Socket(host, port);
  - PrintWriter out = SocketUtils.getWriter(socket);
  - BufferedReader in = SocketUtils.getReader(socket);
    - Slightly longer if you don't use SocketUtils
    - Catch UnknownHostException and IOException
  - Ask for Internet permission in AndroidManifest.xml
- **Formatting and parsing**
  - Format request: printf
  - Handle response
    - Read lines with readLine. Remember it blocks and waits for either end-of-line (String) or closed connection (null).
    - Parse the line with StringTokenizer or String.split
      - To use the split method, learn regular expressions

© 2011 Marty Hall

Questions?

Customized Java EE Training: http://courses.coreservlets.com/
Servlets, JSP, JSF 2.0, Java 6, Ajax, jQuery, GWT, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.