# Programming Assignment 4: Proxy Cache

In this lab you will develop a small web proxy server which is also able to cache web pages. This is a very simple proxy server which only understands simple GET-requests, but is able to handle all kinds of objects, not just HTML pages, but also images.

## Code

The code is divided into three classes as follows:

- `ProxyCache` holds the start-up code for the proxy and code for handling the requests.
- `HttpRequest` contains the routines for parsing and processing the incoming requests from clients.
- `HttpResponse` takes care of reading the replies from servers and processing them.

Your work will be to complete the proxy so that it is able to receive requests, forward them, read replies, and return those to the clients. You will need to complete the classes `ProxyCache`, `HttpRequest`, and `HttpResponse`. The places where you need to fill in code are marked with `/* Fill in */`. Each place may require one or more lines of code.

***NOTE:*** As explained below, the proxy uses DataInputStreams for processing the replies from servers. This is because the replies are a mixture of textual and binary data and the only input streams in Java which allow treating both at the same time are DataInputStreams. To get the code to compile, you must use the -deprecation argument for the compiler as follows:

```
        javac -deprecation *.java
```

If you do not use the -deprecation flag, the compiler will refuse to compile your code!

## Running the Proxy

Running the proxy is as follows:

```
       java ProxyCache port
```

where *port* is the port number on which you want the proxy to listen for incoming connections from clients.

## Configuring Your Browser

You will also need to configure your web browser to use your proxy. This depends on your browser. In Internet Explorer, you can set the proxy in "Internet Options" in the Connections tab under LAN Settings. In Netscape (and derived browsers, such as Mozilla), you can set the proxy in Edit->Preferences and then select Advanced and Proxies.

In both cases you need to give the address of the proxy and the port number which you gave when you started the proxy. You can run the proxy and browser on the same computer without any problems.

# Proxy Functionality

The proxy works as follows.

1. The proxy listens for requests from clients
2. When there is a request, the proxy spawns a new thread for handling the request and creates an HttpRequest-object which contains the request.
3. The new thread sends the request to the server and reads the server's reply into an HttpResponse-object.
4. The thread sends the response back to the requesting client.

Your task is to complete the code which handles the above process. Most of the error handling in the proxy is very simple and it does not inform the client about errors. When there are errors, the proxy will simply stop processing the request and the client will eventually get a timeout.

Some browsers also send their requests one at a time, without using parallel connections. Especially in pages with lot of inlined images, this may cause the page to load very slowly.

## Caching

Caching the responses in the proxy is left as an optional exercise, since it demands a significant amount of additional work. The basic functionality of caching goes as follows.

1. When the proxy gets a request, it checks if the requested object is cached, and if yes, then returns the object from the cache, without contacting the server.
2. If the object is not cached, the proxy retrieves the object from the server, returns it to the client, and caches a copy for future requests.

In practice, the proxy must verify that the cached responses are still valid and that they are the correct response to the client's request. You can read more about caching and how it is handled in HTTP in RFC 2068. For this lab, it is sufficient to implement the above simple policy.

# Programming Hints

Most of the code you need to write relates to processing HTTP requests and responses as well as handling Java sockets.

One point worth noting is the processing of replies from the server. In an HTTP response, the headers are sent as ASCII lines, separated by CRLF character sequences. The headers are followed by an empty line and the response body, which can be binary data in the case of images, for example.

Java separates the input streams according to whether they are text-based or binary, which presents a small problem in this case. Only DataInputStreams are able to handle both text and binary data simultaneously; all other streams are either pure text (e.g., BufferedReader), or pure binary (e.g., BufferedInputStream), and mixing them on the same socket does not generally work.

The DataInputStream has a small gotcha, because it is not able to guarantee that the data it reads can be correctly converted to the correct characters on every platform (DataInputStream.readLine() function). In the case of this lab, the conversion usually works, but the compiler will flag the DataInputStream.readLine()-method as deprecated and will refuse to compile without the -deprecation flag.

It is highly recommended that you use the DataInputStream for reading the response.

## Optional Exercises

When you have finished the basic exercises, you can try the following optional exercises.

1. Better error handling. Currently the proxy does no error handling. This can be a problem especially when the client requests an object which is not available, since the "404 Not found" response usually has no response body and the proxy assumes there is a body and tries to read it.
2. Support for POST-method. The simple proxy supports only GET-method. Add support for POST, by including the request body sent in the POST-request.
3. Add caching. Add the simple caching functionality described above. You do not need to implement any replacement or validation policies. Your implementation will need to be able to write responses to the disk (i.e., the cache) and fetch them from disk when you get a cache hit. For this you need to implement some internal data structure in the proxy to keep track of which objects are cached and where they are on disk. You can keep this data structure in main memory; there is no need to make it persist across shutdowns.