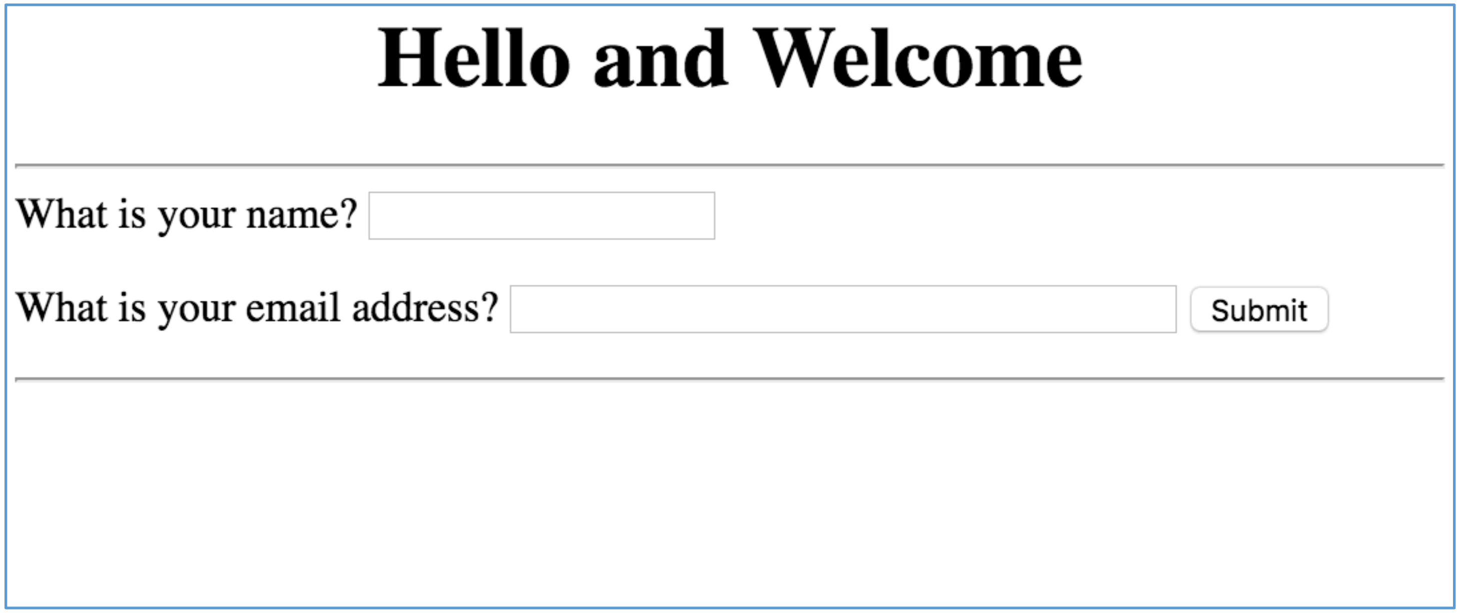
Lez02c - Elis - IT Architect

Java CGI

Ing. Alberto Bucciero

HTML FORM dynamic processing

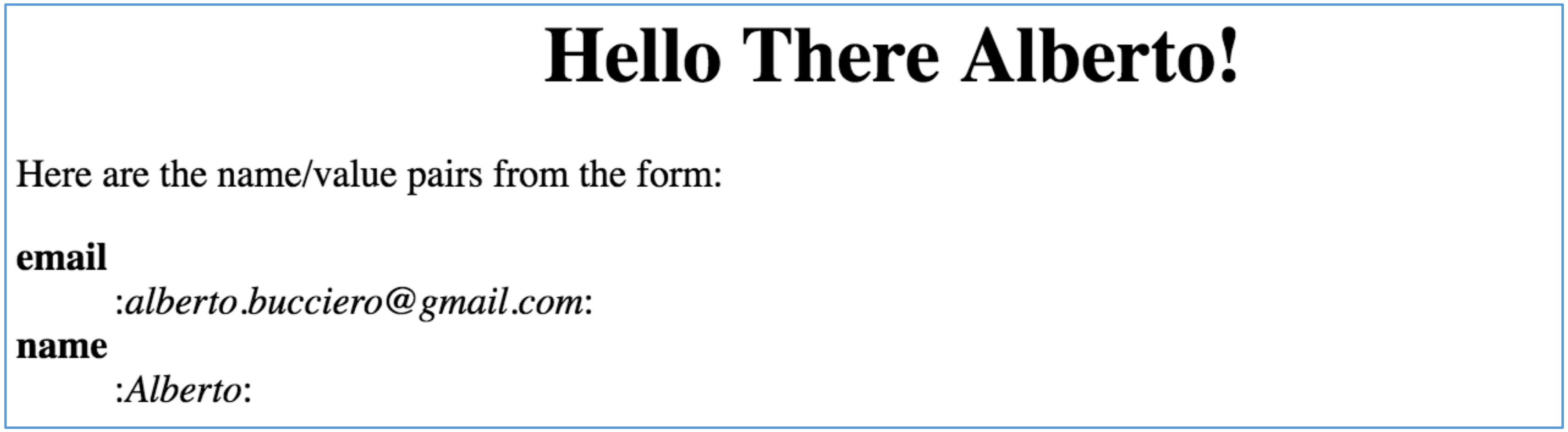


Hello and Welcome

What is your name?

What is your email address?

Result Wanted



Hello There Alberto!

Here are the name/value pairs from the form:

email
:*alberto.bucciero@gmail.com*:
name
:*Alberto*:

- Problem:
  - Often static web pages are insufficient
- Often web server must generate web pages dynamically
  - Based upon parameters supplied by browser
  - Using data fetched from a database
- One solution:
  - Common Gateway Interface (CGI) protocol

CGI

- The Common Gateway Interface (CGI) is a standard for writing programs that can interact through a Web server with a client running a Web browser.
- These programs allow a Web developer to deliver dynamic information (usually in the form of HTML) via the browser.
- A CGI program can be written in any language, including Java, that can be executed by your Web server.
- CGI programs are commonly used to add search engines, guest-book applications, database-query engines, interactive-user forums, and other interactive applications to Web sites.

CGI (2)

- In very basic terms, a CGI program must interpret the information sent to it, process the information in some way, and generate a response that will be sent back to the client.
- Most of the input to a CGI program is passed into it through environment variables.
- I will demonstrate how to send these environment variables to a Java CGI program.
- The rest of the input (if any) is passed into a CGI program as standard input that can be read directly by your program.

URL and URL encoding

- URL format:
  - protocol://host:port/file.cgi?name1=value1&name2=value2&…
- Names and values are URL encoded
- Each special character (", ', =, &, etc.) is encoded as %nn (hex)
- Each space is encoded as +
- Many standard libraries provide functions/methods to encode and decode... (es. **cgi_lib.java**)

HTML Links

```
<a href="http://host:port/file.cgi?name1=value1&name2=value2">
```

- Browser requests that the program named file.cgi be executed
- Web server passes name/value pair(s) to the program

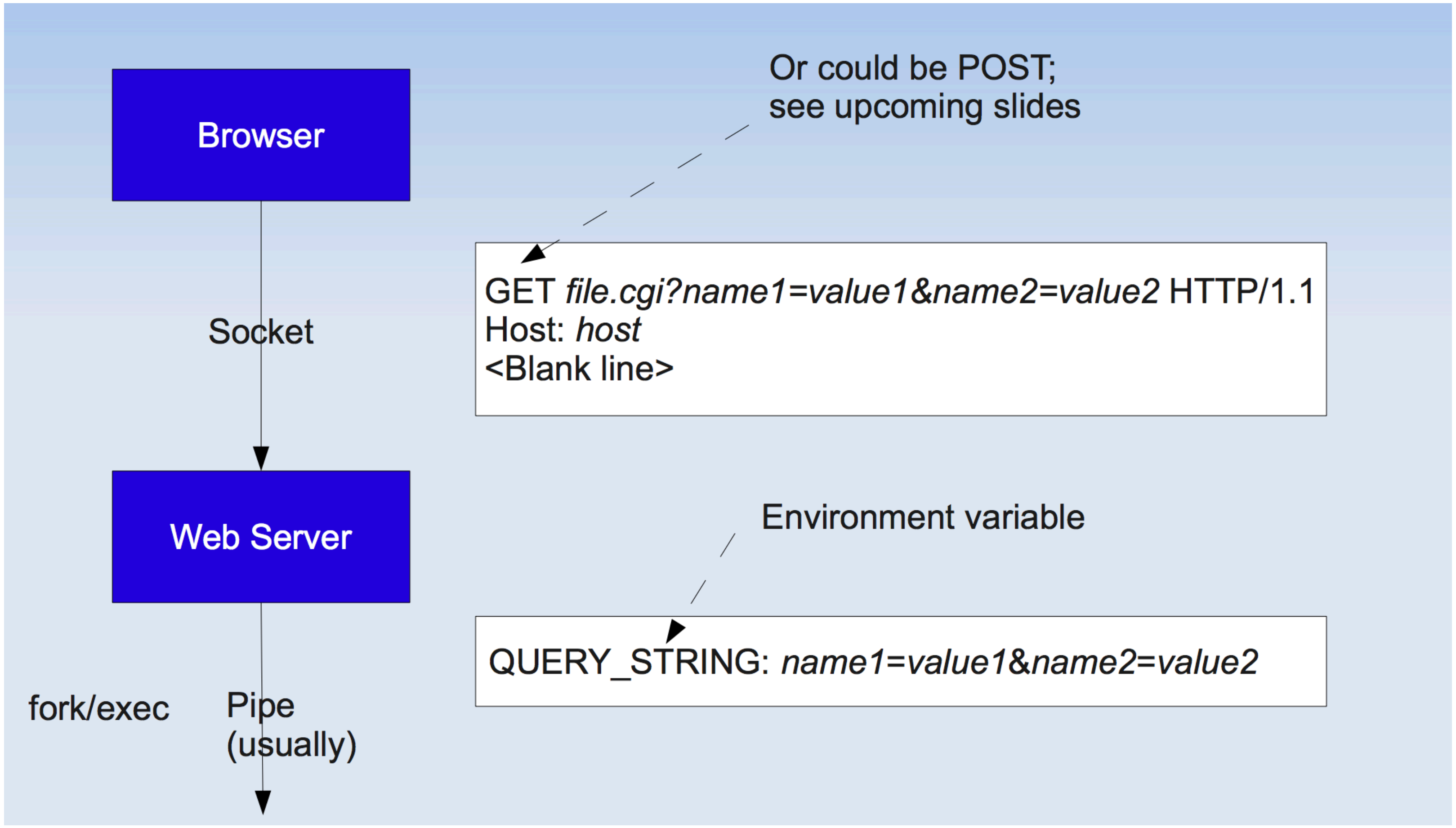
HTML Forms

It could be also POST

```
<form action="http://host:port/file.cgi" method="get">
   <input type="text" name="name1" value="value1"><br>
   <input type="text" name="name2" value="value2"><br>
   <input type="submit">
</form>
```

- Browser requests that the program named file.cgi be executed
- Web server passes name/value pair(s) to the program

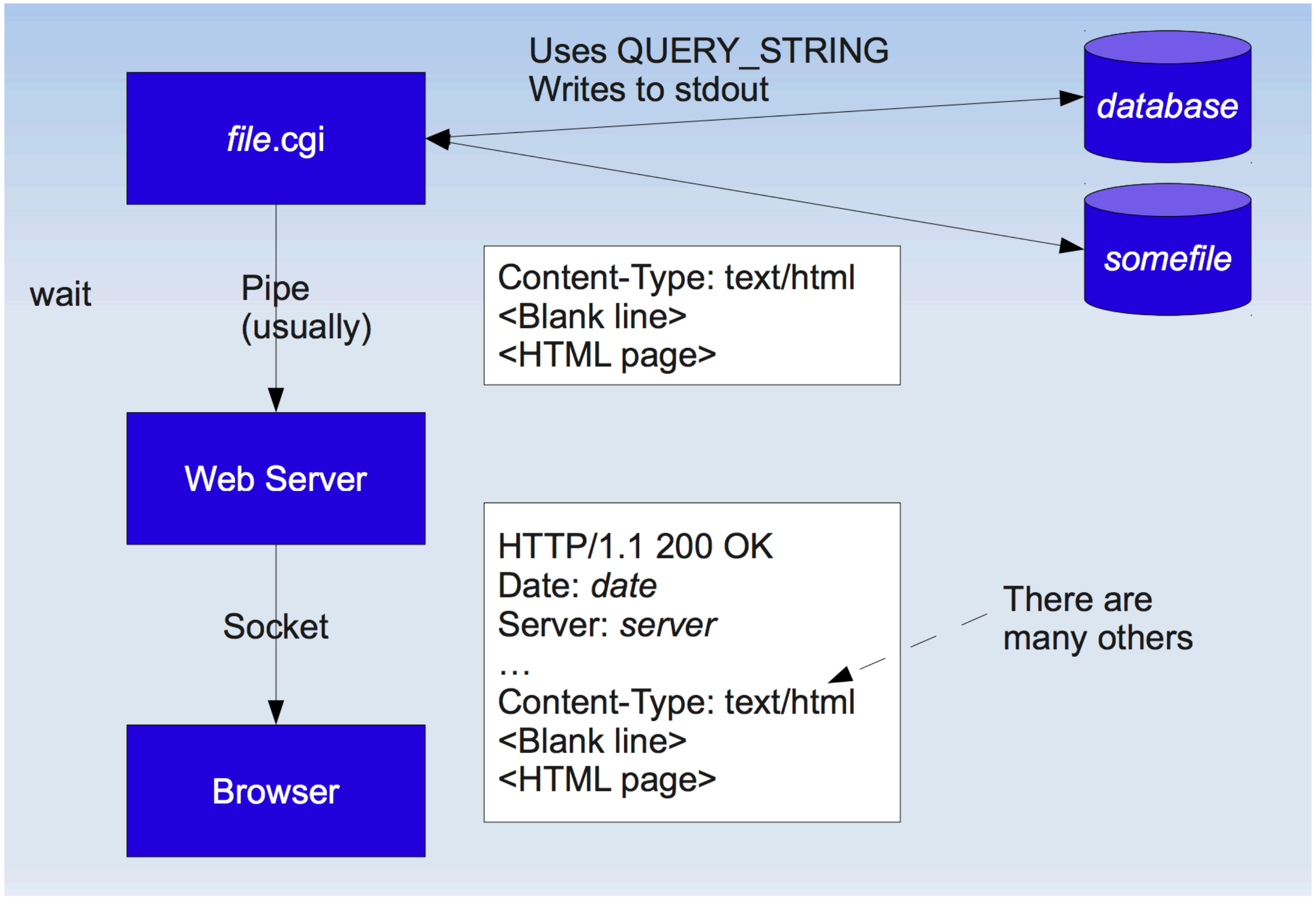
CGI Details: GET Method



GET *file.cgi?name1=value1&name2=value2* HTTP/1.1
Host: *host*
<Blank line>

QUERY_STRING: *name1=value1&name2=value2*

CGI Details: GET Method



See HelloGetJava Application

- index.html
  - User form calls CGI
- hello.cgi
  - Calls "java Hello"
- Hello.java
  - Fetches "person=handle" from QUERY_STRING env var
    Composes response page

- Problem:
  - URL cannot specify execution of Java interpreter
- Solution:
  - Web server executes Bash "helper" script   Bash helper script executes Java interprete

hello.cgi

- The following example shows a Unix script file called hello.cgi invoking a Java program called hello. Note that the -D command-line parameter passes the CGI environment variables into the Java program:
- *#!/bin/sh* java -**Dcgi**.content_type=$CONTENT_TYPE -**Dcgi**.content_length=$CONTENT_LENGTH -**Dcgi**.request_method=$REQUEST_METHOD -**Dcgi**.query_string=$QUERY_STRING -**Dcgi**.server_name=$SERVER_NAME -**Dcgi**.server_port=$SERVER_PORT -**Dcgi**.script_name=$SCRIPT_NAME -**Dcgi**.path_info=$PATH_INFO hello

- This solution doesn't work well on the Windows 95 and NT platforms because there may be limits on the number of characters allowed on the command line.
- An alternative approach might be simply to write each of the environment variables and their associated values to a temporary file (with a unique filename, of course). Then, you may pass the name of this file into your Java program and have it read that file and parse out the environment variable/value pairs. Don't forget to delete the temporary file when you're done using it!
- **This exercise is left to the reader.**

CGI header

- Since a CGI program can return a myriad of document types, a CGI program must place a short header (ASCII text) on its output so that the client will know how to interpret the information it generates.
- Most commonly, CGI programs generate HTML.
- Following the header, a CGI program simply generates the body of the output in its native form.

Passing the CGI environment into the Java program

- Writing a CGI program in Java is fairly easy to do once you understand the issues.
- First and foremost, you need to wrap the execution of the Java program inside another script.
- So, the actual script invoked on your Web server will be a Unix shell script or a Windows batch file (or equivalent) that simply passes the CGI environment variables into your Java program.

Passing the CGI environment into the Java program (2)

- Since Java no longer provides a method to access environment variables directly
(the **System.getenv()** method has been disabled),
- we can pass each CGI environment variable into the Java program using the **-D command-line parameter** on the Java interpreter.

A Java CGI library

- We well use a library of functions including one that generates the appropria te header for HTML.
- It assumes that you have used the approach described above;
- it uses the **System.getProperty()** method to access those command-line parameters.
- If your program needs to use any of the CGI environment variables, you can access them the same way.
- For example, if you want to access the SERVER_NAME environment variable, you could do so as follows:
- **String** server_name = **System**.getProperty("cgi.server_name");

- To ease the tedious task of processing the CGI inputs, there’s a Java class (really a library of functions) that you can utilize to cut down on some of the dirty work.
- This library attempts to duplicate the functionality in the very popular **Perl cgi-lib.pl** library.
- The class is documented the code below using javadoc-style comments so that you can generate HTML documentation directly from the code. (Use **javadoc –private cgi_lib.java** to generate cgi_lib.html.)

Writing your first Java CGI program

- Here's an example that shows how the cgi_lib.java library can be used to write a CGI program.
- We'll write a simple program that processes my "Hello There" form.
- This simple form will prompt the user for a name and email address. Here is the form (hello.html) that we want to process:

Index.html

```
<HTML>
<HEAD>
       <TITLE>Hello and Welcome!</TITLE>
</HEAD>
<BODY>
       <H1 ALIGN=CENTER>Hello and Welcome</H1>
       <hr>
       <FORM METHOD="POST" ACTION="/cgi-bin/hello.cgi">
              What is your name?
              <INPUT TYPE="text" NAME="name" VALUE=""><p>
              What is your email address?
              <INPUT SIZE=40 TYPE="text" NAME="email" VALUE="">

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

Hello.java

- Let's write a Java program to process the "Hello There" form.
- First, we need to let the client know that our program will be generating HTML.
- The Header() method in cgi_lib.java creates the string we need, so we'll start by calling that method and sending the string to standard out using the System.out.println system call.

-
```
// // Print the required CGI header. //
System.out.println(cgi_lib.Header());
```

- Second, we want to process the form data sent to us by the browser. (HTTP HEADER!!!)
- The **ReadParse method** in **cgi_lib.java** does all that work for us and returns the result in an instance of a Hashtable. In this case, the Hashtable will contain two key values after parsing the form data. One will be the "name" input field and the other will be the "email" input field. The values associated with each of these keys will be whatever the user typed into those input fields on the "Hello There" form.

- `// // Parse the form data into a Hashtable. // Hashtable form_data = cgi_lib.ReadParse(System.in);`

- Now that we've parsed the form data, we can do whatever processing we'd like with the data sent to us.

- Then we can generate some HTML to send back to the user's browser.
- In this simple program, we aren't going to do any processing with the data; we're simply going to echo back the information supplied by the user.
- We are going to use the get method on the Hashtable object to extract the form values into strings that we can use in our program.
- The following example shows how we would extract the name that the user typed into a String object.

- `String name = (String)form_data.get("name");`

hello.java

- Now, let's put this all together in a simple program.
- Here is a Java applicationthat we can use to process the "Hello There" form (hello.java):

```java
import java.util.*;
import java.io.*;
class hello
{
  public static void main( String args[] )
  {
      //
      //  Here is a minimalistic CGI program that uses cgi_lib
      //
      //
      //  Print the required CGI header.
      //
      System.out.println(cgi_lib.Header());
      //
      //  Parse the form data into a Hashtable.
      //
      Hashtable form_data = cgi_lib.ReadParse(System.in);
      //
      // Create the Top of the returned HTML page
      //
      String name = (String)form_data.get("name");
      System.out.println(cgi_lib.HtmlTop("Hello There " + name + "!"));
      System.out.println("&lth1 align=center&gtHello There " + name +
                         "!</h1>");
      System.out.println("Here are the name/value pairs from the form:");
      //
      //  Print the name/value pairs sent from the browser.
      //
      System.out.println(cgi_lib.Variables(form_data));
      //
      //  Print the Environment variables sent in from the Unix script.
      //
      System.out.println("Here are the CGI environment variables/value p
                         "passed in from the UNIX script:");
      System.out.println(cgi_lib.Environment());
      //
      // Create the Bottom of the returned HTML page to close it cleanly
      //
      System.out.println(cgi_lib.HtmlBot());
   }
}
```

Notes

- CGI files have to be set to a permission of ’705' (chmod 705)
- Java classes have to be compiled and put in cgi-bin directory
- Java interpreter have to be in PATH env variable

Esercizio

- Creare un’aplicazione composta da:
- Index html: form con INPUT nome e cognome e SUBMIT verso CGI
- CGI chiama una classe JAVA che determina il TIME attuale calcola il tempo mancante rispetto alla
  - 1 pausa
  - 2 pausa
  - Fine della lezione
- Il tutto deve essere opportunamente formattato in HTML