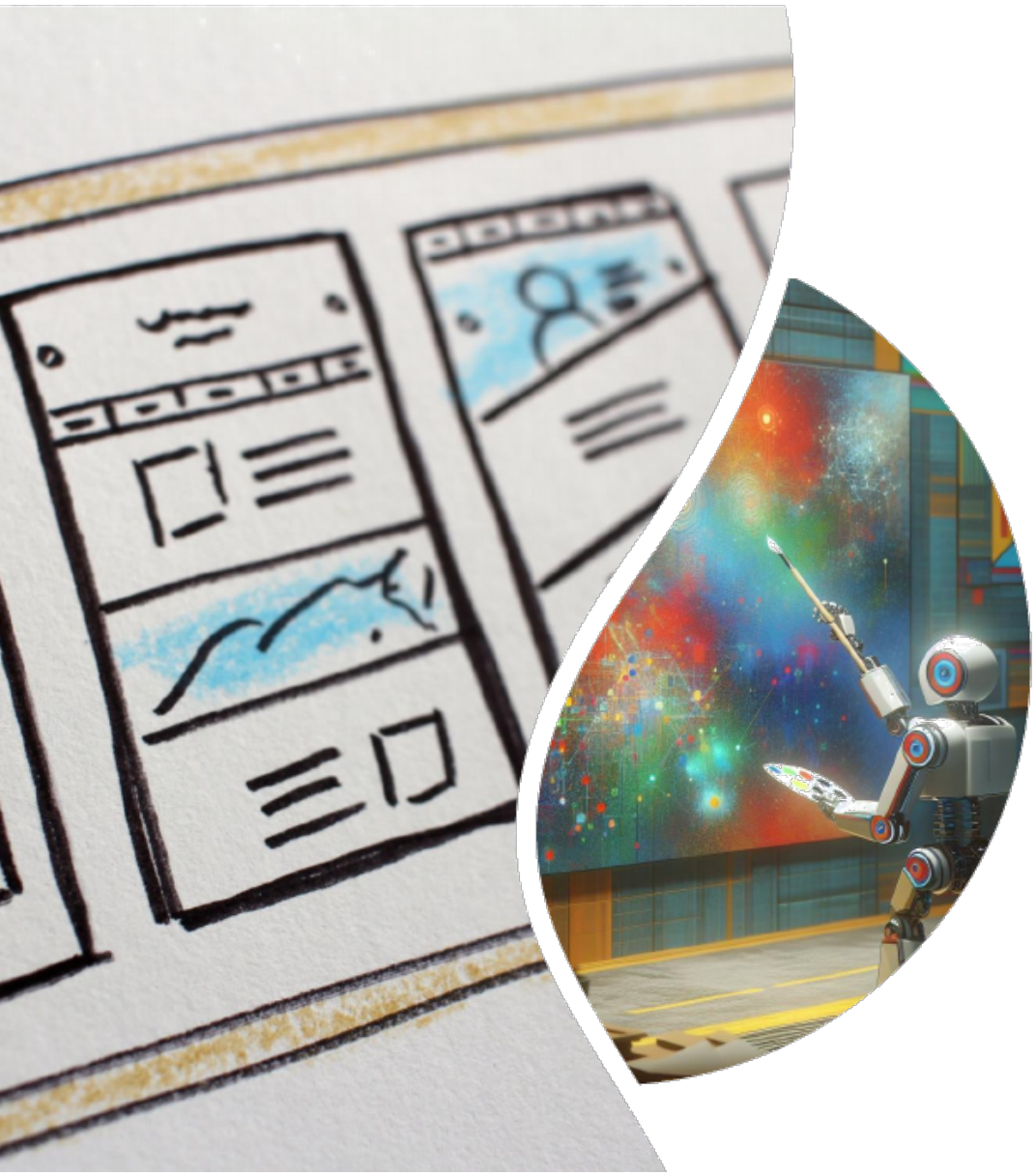

DGIN 5201
Digital Transformation
Lecture 9

Lab 3: Back-end Processing

Tymon Wranik-Lohrenz

Time and date:
11:35–12:25 and
13:05–13:55, 24-Jan-2025
Location: Goldberg CS
134 and 143

Image: DALL-E. Bing Image Creator. Generated by AI

Example e5: Backend Server Processing using CGI

- Using `rsync` copy `e4` to `e5`
- Let us first check that CGI scripts are working by creating file `test.cgi` in e5 as follows:

```
#!/usr/bin/perl
use CGI qw/:standard/;

print header;
print "<html><body>Test\n";
```

- The file should be user executable, without permissions to the group and others (`rwx------`)
- Run the command `./test.cgi` and you should get a simple output as follows:

```
Content-Type: text/html; charset=ISO-8859-1

<html><body>Test
```

- Check in browser: `https://web.cs.dal.ca···dgin5201/e5/test.cgi`

Example e5: Preparing form for processing

- Modify `index.html` the table part:

```
<form method="post" action="register.cgi">
<table>
<tr><th align=right>First and last name:</th>
<td><input type="text" name="name"></td></tr>
<tr><th align=right>Email:</th>
<td><input type="text" name="email"></td></tr>
<tr><th>Certificate (DB, HI, DS):</td>
<td><select name="certificate">
  <option>DB</option><option>HI</option>
  <option>DS</option></select></td></tr>
<tr><td align=center colspan=2>
<input type="submit" value="Submit"/></td></tr>
</table>
</form>
```

Example e5: Processing Data

- Prepare user executable file `register.cgi`:

```
#!/usr/bin/perl
use CGI qw/:standard/;
print header;
print "<html><body><h1>Registration</h1>\n";
print "<p>The following registration is received:\n";
$name  = param('name');  $email = param('email');
$certificate  = param('certificate');
print <<"EOT";
<table>
<tr><th align=right>First and last name:</th>
<td>$name</td></tr>
<tr><th align=right>Email:</th><td>$email</td></tr>
<tr><th>Certificate (DB, HI, DS):</td><td>$certificate
</td></tr><tr><td align=center colspan=2>
<a href="index.html">Back to Registration Page</a></td>
</tr></table>
EOT
```

Example e5: Processing Data and Testing

- Submit some registrations and make sure `register.cgi` works well
- This completes Example e5

Concepts Review: Example 5

- Server-side processing, concept of CGI (Common Gateway Interface)
- Perl programming language, Perl with CGI
- `<form method="post" action="...">`
- `<input ... name="x">`
- `<input type="submit" value="Submit/>`
- CGI processing in Perl

Example e6: Saving Registration Data: Implementation

- Using `rsync` copy `e5` to `e6`; adjust `.htaccess`
- To save registration, add the following line in the script `register.cgi`:

```
...
$email = param('email');
$certificate = param('certificate');

&save_registration($name, $email, $certificate);

print <<"EOT"; ...
```

- and we add the following function at the end of the program:

```
sub save_registration {
  my ($name, $email, $certificate) = @_;
  open (my $fh, ">>registrations-saved.txt") or die;
  print $fh "\nname: $name\nemail: $email\n".
    "certificate: $certificate\n";
  close($fh);
}
```

Example e6: Saving Registration Data: Testing

- First check syntax: `perl -c register.cgi`
- Test the web site by making several registrations
- Check that registrations are saved in the file `registrations-saved.txt`
- Check permissions of `registrations-saved.txt`
  - If not all-readable, make them all-readable
  - Verify that the file is accessible on the web **(!)**
- Change the permissions of `registrations-saved.txt` to user-only readable and writeable
- Check accessibility on the web; **Lesson learned!**
- Check that the application still works

Concepts Review: Example e6

- Perl subroutine (similar concepts: procedure, function)
- Saving and appending data to a file
- Importance of file permissions
- Possible additional issues to deal with files: concurrency (race conditions), efficiency
- Alternatives: using databases, server or file-based

Example e7: Sending Registration by Email

- Use `rsync` to copy e6 to e7
- Modify the `register.cgi` file as follows by adding a new line:

```
...
&save_registration($name, $email, $certificate);
&send_email($name, $email, $certificate);
...
```

- and add the following subroutine at the end of the file:

```
sub send_email {
  my ($name, $email, $certificate) = @_;
  my $emailmessage = "To: vlado\@dnlp.ca\n".
    "Subject: New registration\n\n".
    "A new registration is received as follows:\n\n".
    "name: $name\nemail: $email\n".
    "certificate: $certificate\n";
  open(my $s, "|/usr/lib/sendmail -ti") or die;
  print $s $emailmessage;
  close($s);
}
```

Example e7: Sending Registration by Email (2)

- IMPORTANT: Instead of string `vlado@dnlp.ca` use your own email
- No not forget to use backslash (\) just before the at-sign (@) in email, as in `vlado\@dnlp.ca` because the string is delimited by double-quotes. Otherwise, Perl will replace `@dnlp` with the value of that array
- Test the program and make sure that you receive email after each registration

Example e7: Received Email

- If everything is implemented correctly, and if it works, you should receive an email similar to:

```
From: "...your name..." <YourCSID@willow.cs.dal.ca>
Date: Tue, 13 Feb 2024 14:59:34 -0400 (AST)
To: your_email@dal.ca
Subject: New registration

A new registration is received as follows:

name: Test Name
email: test-email@cs.dal.ca
certificate: DB
```

Example e8: Testing Other Scripting Languages

- Copy e7 to e8 using rsync
- Update `.htaccess` to use passwords from `e8/.htpasswd`
- Create files index-php.html and index-py.html to use PHP and Python as actions: register.php and register.py
- Implement basic register.php and register-py.cgi to print filled form

Example e8: Testing a PHP Script: register.php

```
<html><head><title>Applicant Registration</title></head>
<body>
<h1>Registration</h1>

<p>The following registration is received:

<table>
<tr><th align=right>First and last name:</th>
<td><?php echo $_POST['name'] ?></td></tr>
<tr><th align=right>Email:</th>
<td><?php echo $_POST['email'] ?></td></tr>
<tr><th align=right>Certificate (DB, HI, DS):</th>
<td><?php echo $_POST['certificate'] ?></td></tr>
<tr><td align=center colspan=2>
<a href="index-php.html">Back to Registration Page</a>
</td></tr></table>
```

Example e8: Testing a Python Script: register-py.cgi

```
#!/usr/bin/python
import cgi
print "Content-type: text/html\n\n"
print "<html><body><h1>Registration</h1>\n";
print "<p>The following registration is received:\n";

form=cgi.FieldStorage()
name  = form.getvalue('name')
email = form.getvalue('email')
certificate  = form.getvalue('certificate')
print """<table><tr><th align=right>First and last name:</th>
<td>"""+name+"""</td></tr>
<tr><th align=right>Email:</th><td>"""+email+"""</td></tr>
<tr><th>Certificate (DB, HI, DS):</th>
<td>"""+certificate+"""</td></tr>
<tr><td align=center colspan=2>
<a href="index-py.html">Back to Registration Page</a></td>
</tr></table>\n"""
```

Example e8: Renaming Python Script to `register.py`

- We can copy `register-py.cgi` to `register.py` and try if it works (use `index-py2.html` as the index page)
- It does not! (i.e.., probably does not)
- Solution: Add the following line to `.htaccess` file:

```
AddHandler cgi-script .py
```