The web service we want to provide maintains information on Computer Science students at the University of Liverpool.

The web service should allow us to:

1. Retrieve information on all students
2. Retrieve information on a specific student (via student id)
3. Add information on a new student
4. Modify information on an existing student
5. Delete (information on) a student
6. Partial information on a student
7. Retrieve information on all students

To handle these tasks, we need a server-side program or programs that deal with all these HTTP requests. The web service should allow us to:

- Use a server-side program named `REST.php`
- Have just one PHP script named `REST.php` in the directory corresponding to the base URL of the web service
- Using the rewriting facility of the web server, rewrite URIs so that a resource becomes a trailing pathname
- The script can use `$_SERVER['PATH_INFO']` to determine the trailing pathname and thereby the resource that a request refers to.

The scripts would all be identical and use `$_SERVER['SCRIPT_URI']` to determine the corresponding resource:

- thousands of identical scripts
- very difficult to maintain all the scripts
- directory structure changes as resources are added and deleted

Option 1: Multitude of scripts
- For URIs where the last URI component maps to a directory, web servers will typically look for a file `index.html` or `index.php` in that directory
- We could create a directory hierarchy corresponding to the resource hierarchy
- In each directory we place a PHP script `index.php` that deals with requests relating to the corresponding resource
- The script would all be identical and use `$_SERVER['SCRIPT_URI']` to determine the corresponding resource

Option 2: One script plus URI rewriting to trailing pathnames
- We have just one PHP script with an arbitrary name, for example, `REST.php`, in the directory corresponding to the base URL of the web service
- Using the rewriting facility of the web server, we rewrite URIs so that a resource becomes a trailing pathname

Option 3: One script plus URI rewriting to query
- We have just one PHP script with an arbitrary name, for example, `REST.php`, in the directory corresponding to the base URL of the web service
- Using the rewriting facility of the web server, we rewrite URIs so that a resource becomes (part of) a query

Option 4: One script plus URI in query
- We have just one PHP script with an arbitrary name, for example, `REST.php`, in the directory corresponding to the base URL of the web service
- The resource must be passed to the script via a query parameter, for example, `resource`
- The script can use `$_REQUEST['resource']` to determine the trailing pathname and thereby the resource that a request refers to

Rewrite rules are not always enabled,
Trailing pathnames are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
might require administrator rights for the web server

Rewrite rules are not always enabled,
Web server configuration and .htaccess Files

- **Rewrite rules** for URLs are part of the web server configuration.
- .htaccess files provide a way to make configuration changes to a web server on a per-directory basis.
- .htaccess files contain configuration directives.
- Rewrite rules are one of many kinds of configuration directives.
- The directives apply to directory containing the .htaccess file and all its subdirectories.
- Directives are applied in the order that they are found.

On our web server .htaccess files need to be world-readable.

URL Rewrite Rules

The general format of rewrite rules is:

```
RewriteRule pattern substitution [flags]
```

- **pattern** is a Perl compatible regular expression
- **pattern** may contain capture groups
- **pattern** is matched relative to the current directory
  - **not** the original, full URL in the request
  - **not** the part of original URL-path that it matched
- **substitution** is a string that replaces the original URL-path that was matched by **pattern**
  - **not** the part of original URL-path that it matched
- **substitution** may contain back-references ($0) to **pattern**
- **flags** is a comma-separated list of flags, modifying the behaviour of the rewrite rule

Example rewrite rules:

```
RewriteRule b/ file1.html file2.htm
RewriteRule ^b/(.*) file2.php?path=\1
```

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>b/file1.html</td>
<td>file2.htm</td>
</tr>
<tr>
<td>^b/(.*)</td>
<td>file2.php?path=\1</td>
</tr>
</tbody>
</table>

mapped path becomes

- `b/file1.html`
- `file2.php?path=\1`

Forces the resulting URI to be passed back to the URL mapping engine for processing.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Escape/encode non-alphanumeric characters in backreferences before applying the transformation</td>
</tr>
<tr>
<td>BNF</td>
<td>If backreferences are being escaped/encoded, spaces should be escaped to %20 instead of +</td>
</tr>
<tr>
<td>QSA</td>
<td>Appends any query string from the original request URL to any query string created in the rewrite target</td>
</tr>
<tr>
<td>QSD</td>
<td>Discard any query string attached to the incoming URI</td>
</tr>
<tr>
<td>RC</td>
<td>Makes the pattern comparison case-insensitive</td>
</tr>
<tr>
<td>L</td>
<td>Stop the rewriting process immediately and do not apply any more rules</td>
</tr>
<tr>
<td>R</td>
<td>Forces an external redirect, optionally with the specified HTTP status code</td>
</tr>
<tr>
<td>PT</td>
<td>Forces the resulting URI to be passed back to the URL mapping engine for processing</td>
</tr>
</tbody>
</table>
URL Rewrite Rules: Flags

- Flags modify the behaviour of rewrite rules

<table>
<thead>
<tr>
<th>Flag</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Escape/encode non-alphanumeric characters in backreferences before applying the transformation</td>
</tr>
<tr>
<td>QSA</td>
<td>Appends any query string from the original request URL to any query string created in the rewrite target</td>
</tr>
</tbody>
</table>

# - sgxyz/public_html/a/.htaccess
RewriteRule 'b'/(.*)/\w+.php$ file2.php?path=$1 [B, QSA]

- Mapped path is - sgxyz/public_html/a/b/c/d/file1.php?sort=true
- Prefix up to directory a/ will be stripped
- Pattern "b'/(.*)/\w+.php$"
- does match b/c/file1.php
- Mapped path becomes - sgxyz/public_html/a/file2.php?path=c/d&sort=true

Example (3)

- The web service we want to provide maintains information on Computer Science students at the University of Liverpool
- The web service should allow us to
  - Add information on a new student
    POST https://api.liv.ac.uk/v1/students
    + information on a new student in the request body
    + ...
- We will use Option 3 to deal with HTTP request to our web service
- We will rewrite HTTP requests so that
  method https://api.liv.ac.uk/v1/resource?query
becomes
  method https://api.liv.ac.uk/v1/REST.php?resource=resource ↪

- The script REST.php can use $_REQUEST['resource'] to determine the resource that a request refers to

Revision and Further Reading

- Read
  - Apache HTTP Server Tutorial: .htaccess files
    https://httpd.apache.org/docs/2.4/howto/htaccess.html
  - Apache Module mod_rewrite
    https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html
  - RewriteRule Flags
    http://httpd.apache.org/docs/current/rewrite/flags.html