CSE 3101: Internet Computing II

Lab Session 9

Paul Crawford

Semester 1, Week 14 (26th & 27th November, 2018)

1. Aims

1.1. Further understanding of Web applications, including client-side Web Forms (with server-side processing upon submission), server-side PHP functions, and Web Service architectures.

1.2. Further increased facility with the ongoing sets of concepts & techniques needed for eventual completion of the Final Project.

1.3. **POLICY CHANGE (REMINDER)**: Please note that henceforth **all Lab Sessions will be marked** and count as a portion of the overall Lab grade (10%). Completed labwork must be zipped into an archive (filename given later below at the end of § 2.1), and submitted to the Tutor (<pcrawford@mac.com>) by midnight Friday each week, with the Subject line: 'CSE3101 - Lab <*N*> - <*YourGroupName*>'.

2. Tasks

2.1. Web Forms, and Service API 'Layer': "Unlocking" Search Capability for Staff

2.1.1. **NOTE: This Lab is a continuation of the previous Lab 08, i.e., its functionality occurs within the context of a complete (albeit very bare-bones) UoG CS Dept. prototype website.**

2.1.2. Please try to ensure that your *x*AMP[P] stack includes a modern version of PHP at least ≥ v5.4.0 (e.g., in XAMPP, by navigating to 'localhost/dashboard', and then clicking 'PHPInfo' in the top bar). This will help to avoid any "missing-feature" errors such as those related to the relatively recent '`http_response_code()`' function. {It might end up being necessary for the Tutor or Lecturer to make formal upgrade requests to the Technical staff (?).}

2.1.3. Continue to use the Lab 08 test database {named 'uog_cs_prototype_db'}, from the last Lab session. NOTE: It is assumed that the DB has already been populated with at least the two test Staff Members from that Lab session (viz., 'Test Tester' and 'Test2 Tester2').

2.1.4. Continue to use the 'uog_cs_prototype' subfolder {located in your Apache Web Root directory (e.g., 'C:\xampp\htdocs')}, from the last Lab session.

2.1.5. **Modify** the ‘C:\xampp\uog_cs_prototype**\site\category\staff.php**’ PHP source file, to allow **(high-level) searching** for matching Staff Members.

I.e., a very simple Search form is to be added at the top of the dynamically-generated Staff (List) Page, and any submission from that form is now to be handled appropriately. As part of this task, the existing Search functionality (currently unused) in the full Staff Class/Object file will be “unlocked”.

Recall that you could use either the '`if`' statement or the ternary conditional operator (`?:`) when checking for an empty GET field-value or object-member, etc. Also, a useful built-in predicate to test for empty values is the aptly-named: `empty()`.

For example, we could implement the Staff Search functionality, as follows (where changes are indicated in **larger bold**, and the section is also now updated to make it clearer that certain existing code is to be left intact and not accidentally replaced, etc.):

```
[…]

// retrieve the submitted query data (if any)
$staff_member->id = ( isset( $_GET[ "id" ] ) ? $_GET[ "id" ] : 0 );
$bGetAll = empty( $staff_member->id );
$search_keywords = ( isset( $_GET[ "s" ] ) ? $_GET[ "s" ] : "" );
$bInSearchMode = ! empty( $search_keywords );

// retrieve the requested data from the database via the Service API, […]
// [NOTE: The existing code has been “re-wrapped” for readability]
// [NOTE: Please do *not* forget to also add the new outermost
//  closing ')' indicated {after the existing '…->readOne() )'}]
$result_data_json = ( $bInSearchMode
                        ? $staff_member->search( $search_keywords )
                        : ( $bGetAll
                              ? $staff_member->read()
                              : $staff_member->readOne() ) );

[…]

// translate that PHP data into markup language
if ( $bInSearchMode || $bGetAll ) {
    $result_data = "<h1>Staff List</h1>";

    // generate the simple Search form at the top
    $result_data .= '<form action="staff.php" method="get">' .
        'Keywords: <input name="s" type="text">' .
        '<input name="submit" type="submit" value="Search">' .
        "</form>" .
        "<br/><hr><br/>";

    // dispatch on processing status
    […]

} else {  // there's just one record
    $result_data = "<h1>Staff Member Details</h1>";

    […]
}

[…]
```

2.1.6. **Modify** the ‘C:\xampp\uog_cs_prototype**\site\object\staff_obj.php**’ PHP source file, to allow **passing-through to lower-level (API) searching** for matching Staff Members.

I.e., within the ‘StaffMemberViaAPI’ Class, the existing commented-out '`search()`' method will now simply be **un-commented**.

For example, we could implement the Staff Search passing-through functionality, as follows (where changes are indicated in **larger bold**):

```
<?php

// include core file
require_once "../../config/core.php";

class StaffMemberViaAPI {

    […]


    // methods:-

    […]

    // [Move comment-block "head" to *after* the 'search()' method]
    ~~// !!! NOTE: The methods below are NOT yet […] Service API !!!~~
    ~~/*~~
    // search Staff members
    function search( $keywords ) {
        [Just leave existing statements as-is …]
    }

    // [This comment-block "head" was moved to here from above]
    // !!! NOTE: The methods below are NOT yet […] Service API !!!
    /*
    // read Staff members with pagination
    public function readPaging( $from_record_num, $records_per_page ) {
        […]
    }

    […]
    */
} // class StaffMemberViaAPI
?>
```

2.1.7. **Modify** the ‘C:\xampp\uog_cs_prototype**\site\site_api\category_api\staff_api.php**’ PHP source file, to allow **lower-level (API) searching** for matching Staff Members.

I.e., the existing Search functionality (currently unused) in the full Staff Class/Object file will be “unlocked”.

Recall that you could use either the '`if`' statement or the ternary conditional operator (`?:`) when checking for an empty GET field-value or object-member, etc. Also, a useful built-in predicate to test for empty values is the aptly-named: `empty()`.

For example, we could implement the Staff Search lower-level (API) functionality, as follows (where changes are indicated in **larger bold**, and the section is also now updated to make it clearer that certain existing code is to be left intact and not accidentally replaced):

```
[…]

// retrieve the submitted query data (if any)
$staff_member->id = ( isset( $_GET[ "id" ] ) ? $_GET[ "id" ] : "" );
$bSetErrRespCodes = ( isset( $_GET[ "set_error_resp_codes" ] )
                        ? (bool) $_GET[ "set_error_resp_codes" ] : true );
$bGetAll = empty( $staff_member->id );
$search_keywords = ( isset( $_GET[ "s" ] ) ? $_GET[ "s" ] : "" );
$bInSearchMode = ! empty( $search_keywords );

// retrieve the requested data from the database, in PDO format
if ( $bInSearchMode ) {
     $result_data_pdo = $staff_member->search( $search_keywords );

} else if ( $bGetAll ) {
    $result_data_pdo = $staff_member->read();

} else {
    $staff_member->readOne();
}

// convert that retrieved [PDO] data into JSON format
if ( $bInSearchMode || $bGetAll ) {
    […]

} else {  // there's just one item
    […]
}

[…]
```

2.1.8. Continue to exercise the now–Search-enhanced Web Forms & Service API functionality (this time, all URIs should be tested directly via any Web browser, **not** a specialised web service such as Postman), and save the various **result files** as follows:

2.1.8.1. Navigate to 'localhost/uog_cs_prototype/**site/**'. This will show the Site Display "Master" Page, with links to all of the Category (List) Pages. Then, click the **Staff** link {to retrieve all the existing Staff Members, indirectly via an internal JSON API layer}. Note that there should already be at least two entries, from the last Lab session (viz., 'Test Tester' and 'Test2 Tester2').

Next, via the simple Search form at the top, submit the following **Search query**:

Keywords : Test2

Then, save the resulting HTML page (there should be just 1 matching entry, for 'Test2 Tester2') to a file named 'Staff_SearchViaAPI_**AfterU_First**.html'.

2.1.8.2. Again on that Staff (List) Page, submit the following **Search query**:

Keywords : Tester

Then, save the resulting HTML page (there should now be at least 2 matching entries, for both ‘Test Tester’ and ‘Test2 Tester2’) to a file named ‘Staff_SearchViaAPI_**AfterU_Second**.html’

2.1.8.3. From the above **3 modified source files** and **2 result files** (**5 files overall**) — i.e.: ‘**staff.php**’, ‘**staff_obj.php**’, ‘**staff_api.php**’, ‘Staff_SearchViaAPI_**AfterU_First**.html’, and ‘Staff_SearchViaAPI_**AfterU_Second**.html‘ — create a zip archive named ‘CSE3101_Lab09_<*YourGroupName*>.zip’ and submit it as described in § 1.3.

2.2. Overall Considerations

2.2.1. For additional information, practice and code samples, you could also explore any of various useful articles, tutorials & references available online. For instance, the **Wikipedia** site (<en.wikipedia.org>) has several detailed articles on Web Services, Web APIs, the REST architecture, etc.