How to Implement a Library

This document explains how to implement support for a library.

Example Library Implementation

Each library implementation has an entry point function called **update** that:

- Navigates to the main OPAC page. This is usually necessary to initialise the session and to get a cookie. ❶
- Log into the OPAC. ❷
- Get the links for each of the page(s) containing the loans and holds information. ❸
- Download and parse the information. ❹

```
      - (BOOL) update
      {
❶ →       URL *logoutURL  = [catalogueURL URLWithPath: @"/cgi-bin/spydus.exe/PGM/OPAC/CCOPT/LB/3?RDT=/spydus.html"];
          URL *accountURL = [catalogueURL URLWithPath: @"/cgi-bin/spydus.exe/MSGTRN/OPAC/LOGINB"];
          [browser go: logoutURL];
          [browser go: accountURL];

❷ →       // Log in
          if ([browser submitFormNamed: @"frmLogin" entries: self.authenticationAttributes] == NO)
          {
              [Debug log: @"Failed to login"];
              return NO;
          }
          [self authenticationOK];

❸ →       URL *loansURL                = [browser linkForLabel: @"Current loans"];
          URL *overdueLoansURL         = [browser linkForLabel: @"Overdue loans"];
          URL *holdsReadyForPickupURL  = [browser linkForLabel: @"Reservations available for pickup"];
          URL *holdsURL                = [browser linkForLabel: @"Reservations not yet available"];

          // Loans
❹ →       [browser go: loansURL];
          [self parseLoans];

          // Overdue loans
          [browser go: overdueLoansURL];
          [self parseLoans];

          // Holds ready for pickup
          [browser go: holdsReadyForPickupURL];
          [self parseHoldsReadyForPickup: YES];

          // Holds not yet available
          [browser go: holdsURL];
          [self parseHoldsReadyForPickup: NO];

          return YES;
      }
```

The loans information is parsed by **parseLoans** which:

- Looks for the first <**table**/> element that contains the keyword **BIBENQ**. ❶
- Analyses and extracts the table to determine the title, author and due date rows. ❷

- Calls **addLoans:** to save the rows. ❸

```
- (void) parseLoans
{
    [Debug log: @"Parsing loans"];
    NSScanner *scanner   = browser.scanner;
    HTMLElement *element = nil;

    [scanner scanPassElementWithName: @"head"];

❶ →    if ([scanner scanNextElementWithName: @"table" regexValue: @"BIBENQ" intoElement: &element])
    {
❷ →        NSArray *columns    = [element.scanner analyseLoanTableColumns];
        NSArray *rows       = [element.scanner tableWithColumns: columns];
❸ →        [self addLoans: rows];
    }
}
```

Parsing the holds information is the same as the loans.

```
- (void) parseHoldsReadyForPickup: (BOOL) readyForPickup
{
    [Debug log: @"Parsing holds"];
    NSScanner *scanner   = browser.scanner;
    HTMLElement *element = nil;

    [scanner scanPassElementWithName: @"head"];

    // Holds ready for pickup
    if ([scanner scanNextElementWithName: @"table" regexValue: @"BIBENQ" intoElement: &element])
    {
        NSArray *columns    = [element.scanner analyseHoldTableColumns];
        NSArray *rows       = [element.scanner tableWithColumns: columns];

        if (readyForPickup) [self addHoldsReadyForPickup: rows];
        else                [self addHolds:               rows];
    }
}
```

The **myAccountURL** method is called when the user selects the **My Account** menu option. It automatically logs the user into their account page and displays it in the web browser.

```
- (URL *) myAccountURL
{
    URL *logoutURL  = [catalogueURL URLWithPath: @"/cgi-bin/spydus.exe/PGM/OPAC/CCOPT/LB/3?RDT=/spydus.html"];
    URL *accountURL = [catalogueURL URLWithPath: @"/cgi-bin/spydus.exe/MSGTRN/OPAC/LOGINB"];
    [browser go: logoutURL];
    [browser go: accountURL];

    return [browser linkToSubmitFormNamed: @"frmLogin" entries: self.authenticationAttributes];
}
```