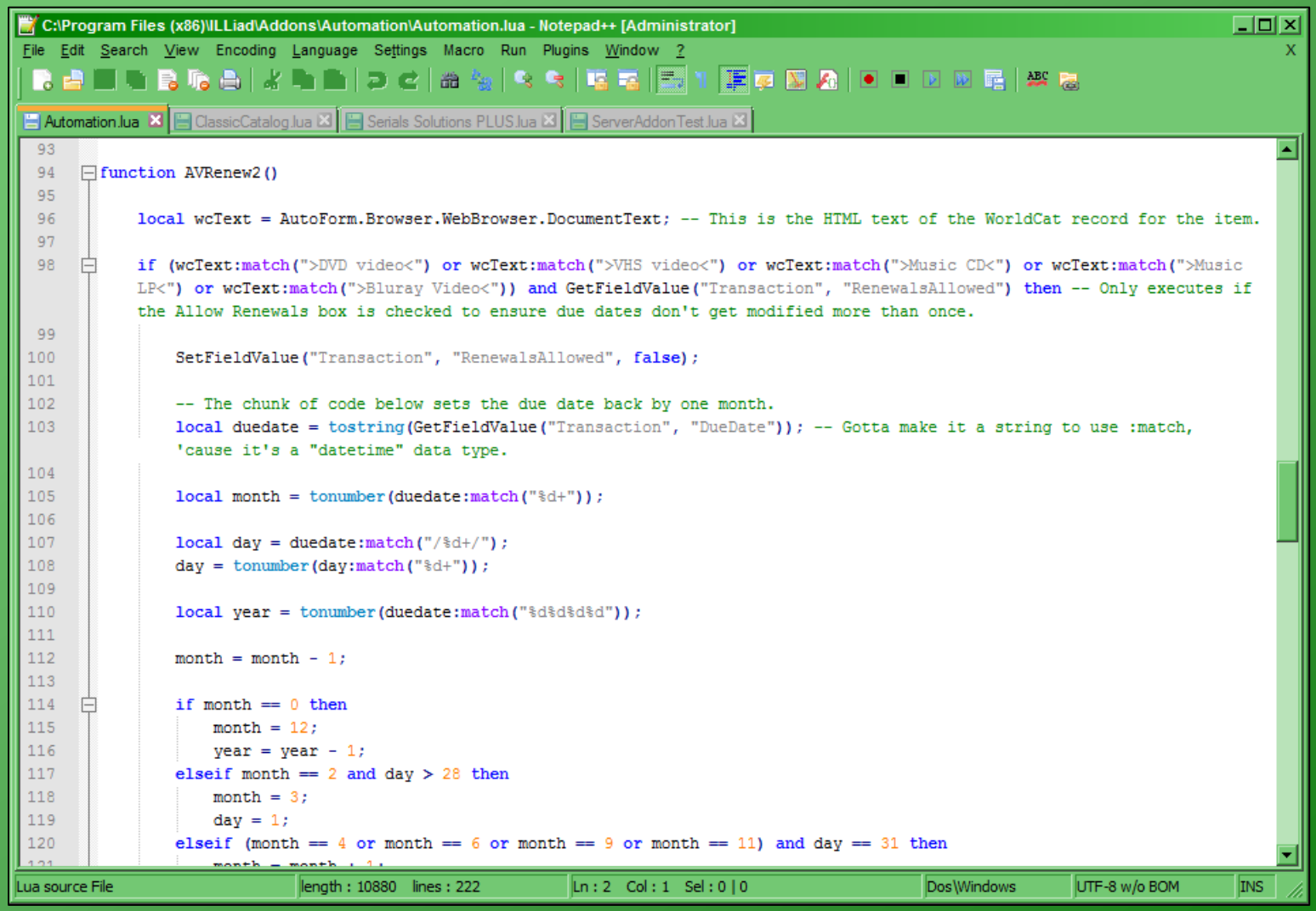
DIY Addons for ILLiad



Andrew Morgan, East Carolina University (ERE)

What This Presentation Covers:

- The basic function and structure of client-side addons
- Resources for creating and editing client-side addons
- Examples of how to use client-side addons to streamline workflows and add functionality

What This Presentation Does Not Cover:

- Server addons (sorry, I only have an hour!)
- System-level addons
- How to become fluent in Lua

Types of Addons

Client

- Runs when a transaction is opened
- Usually, but not always, a tab in the transaction
- Examples: WorldCat Local Search, Serials Solutions Citation Linker, CCC GetItNow

System

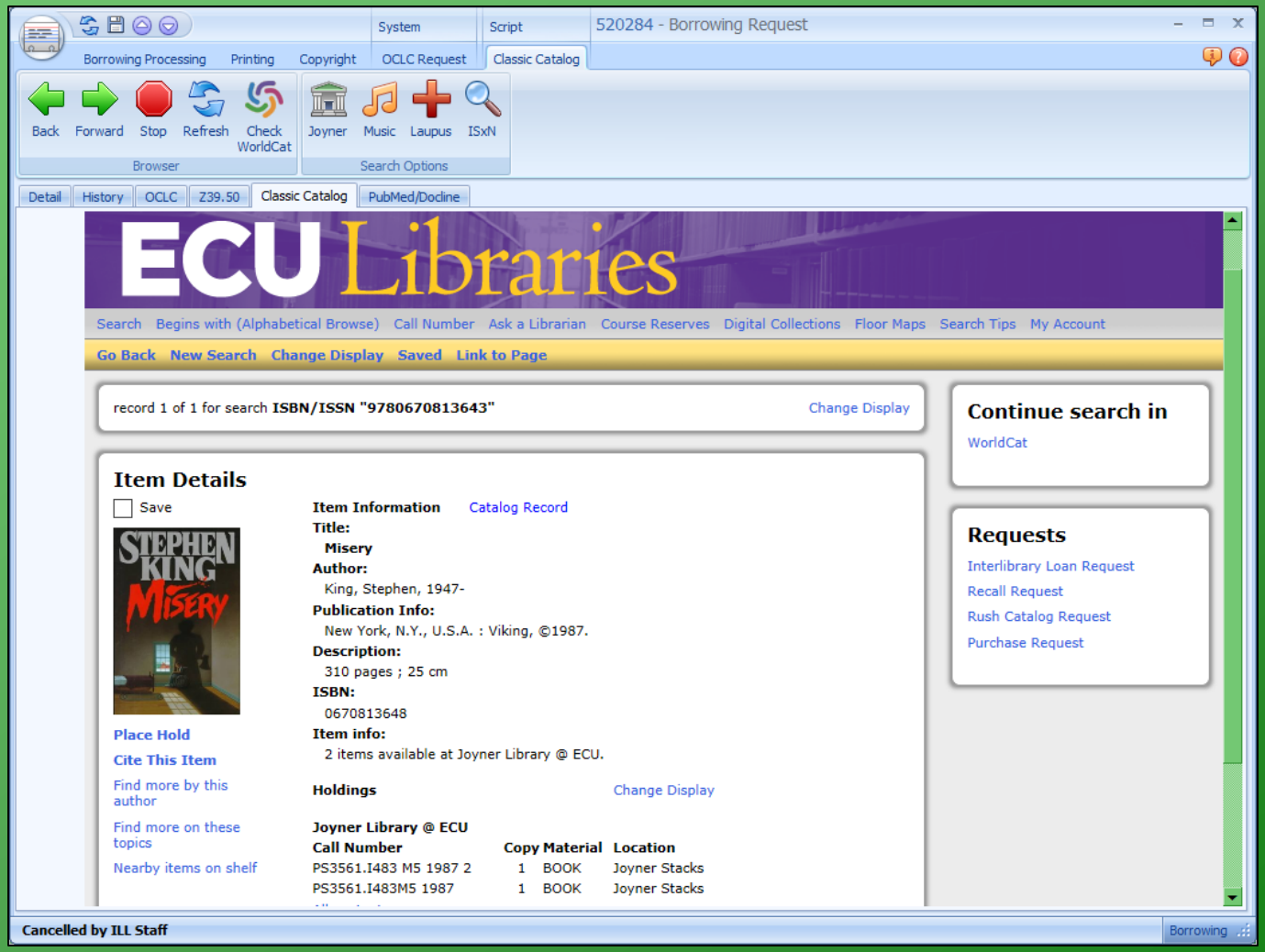
- Usually runs when a select few actions are performed on a transaction (marked found, checked in from lender, etc.)
- Can also be used to display an addon tab on the ILLiad home screen
- Examples: IDS NCIP Client, Workflow Toolkit

Server

- Run at set intervals in the background—requires no manual trigger or intervention.
- Uses SQL queries to affect multiple transactions at once.
- Examples: Electronic Delivery Reminder, Twilio SMS Notifications

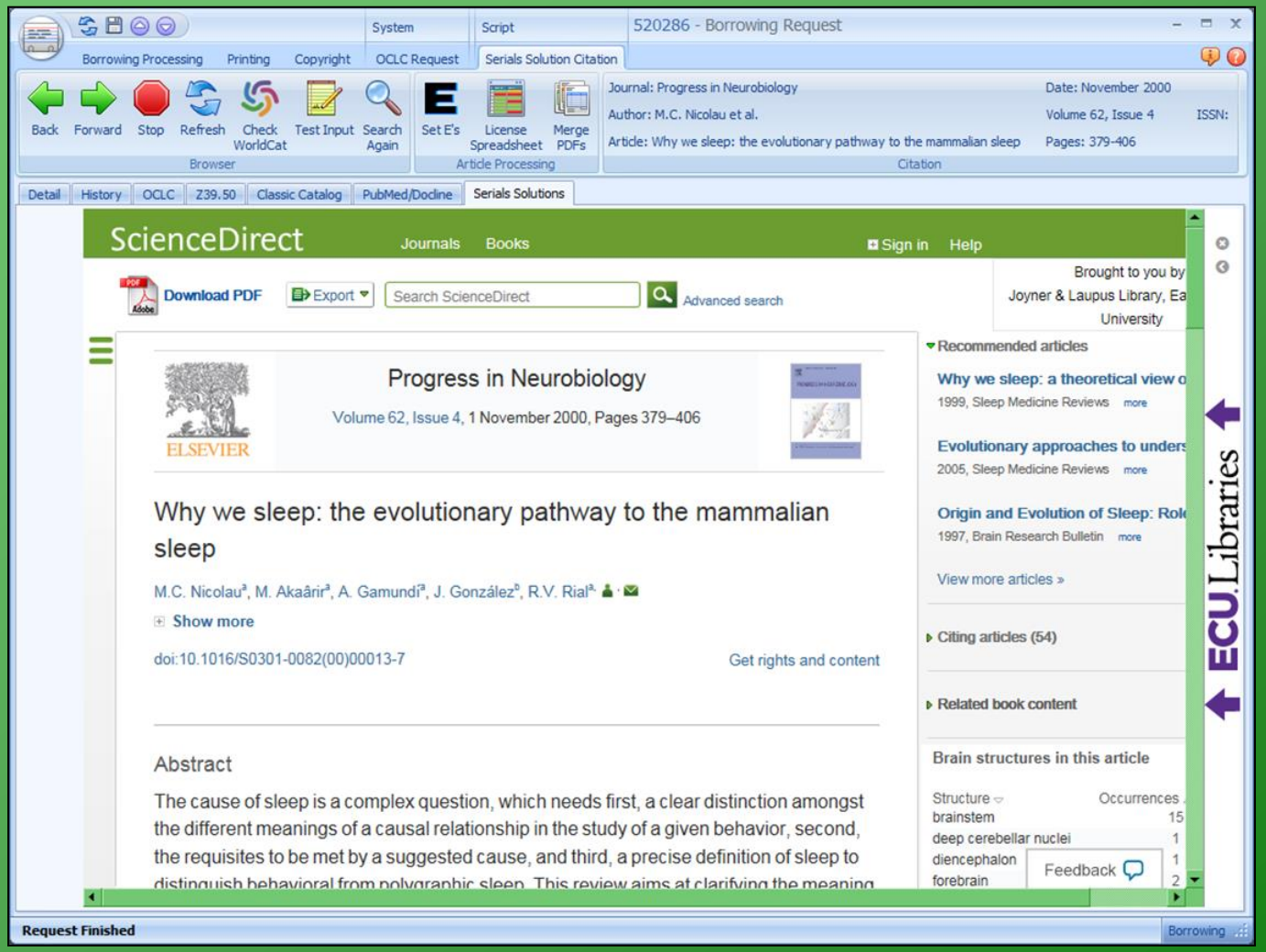
The Possibilities...

Search your library's OPAC automatically!



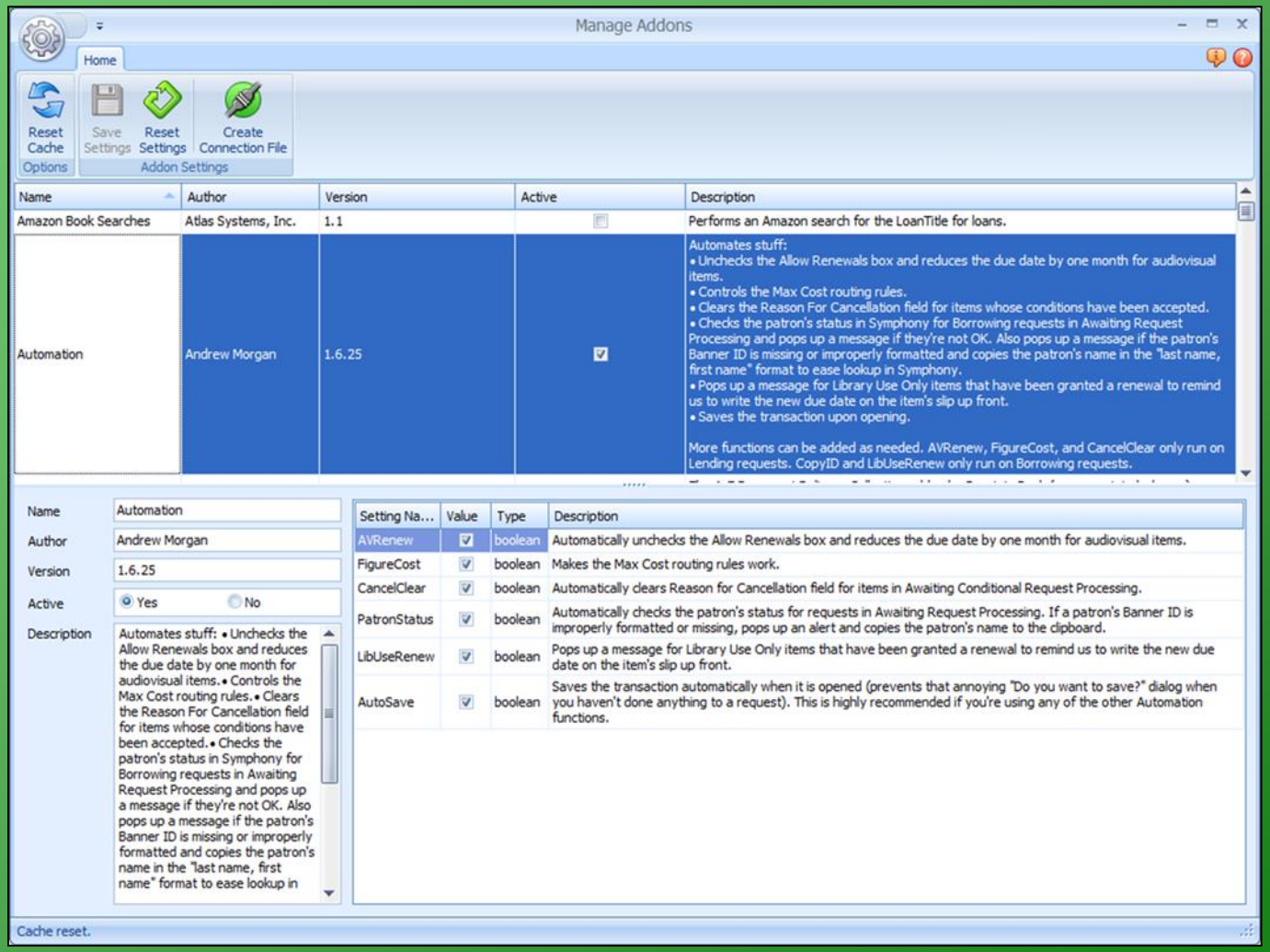
The Possibilities...

Modify existing addons to suit your needs!



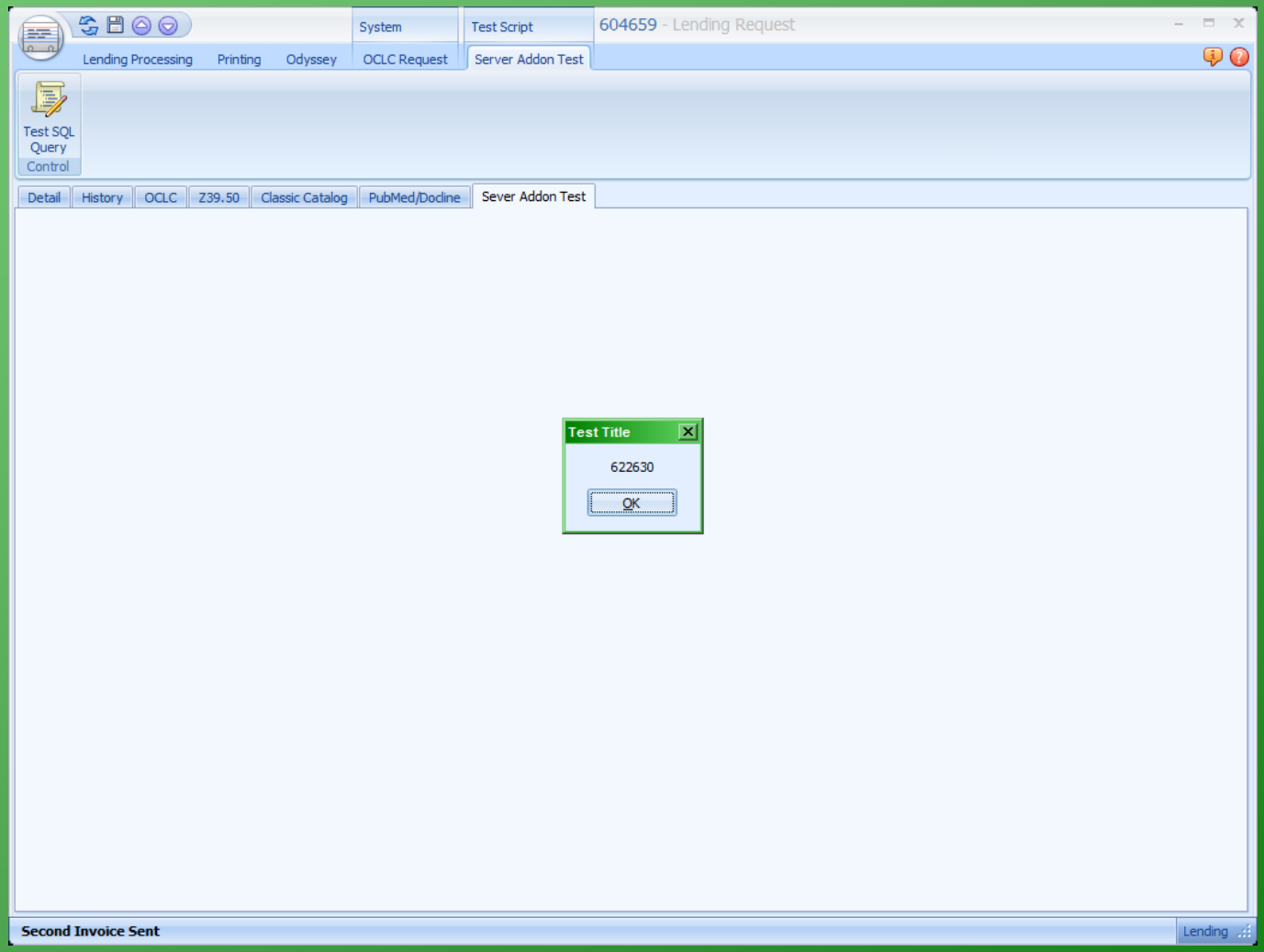
The Possibilities...

Automate processes behind the scenes!



The Possibilities...

Test server addons!



This is more exciting than a still image can convey. Trust me.

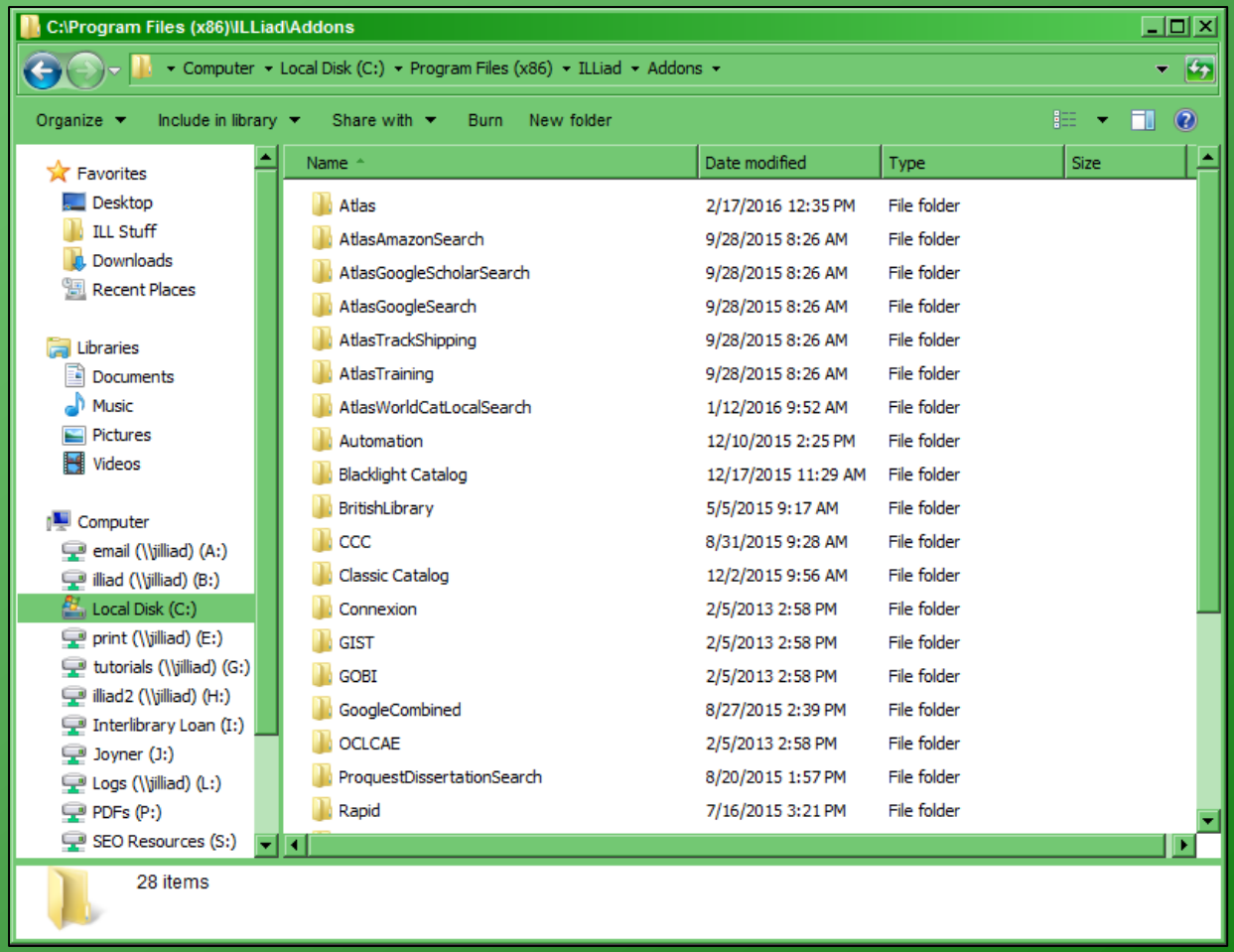
What You'll Need

- ~~¼ cup of butter~~
- Notepad++
- Lua reference manual
  - Lua demo
- ILLiad addons documentation
  - Key data tables
- ILLiad addons directory

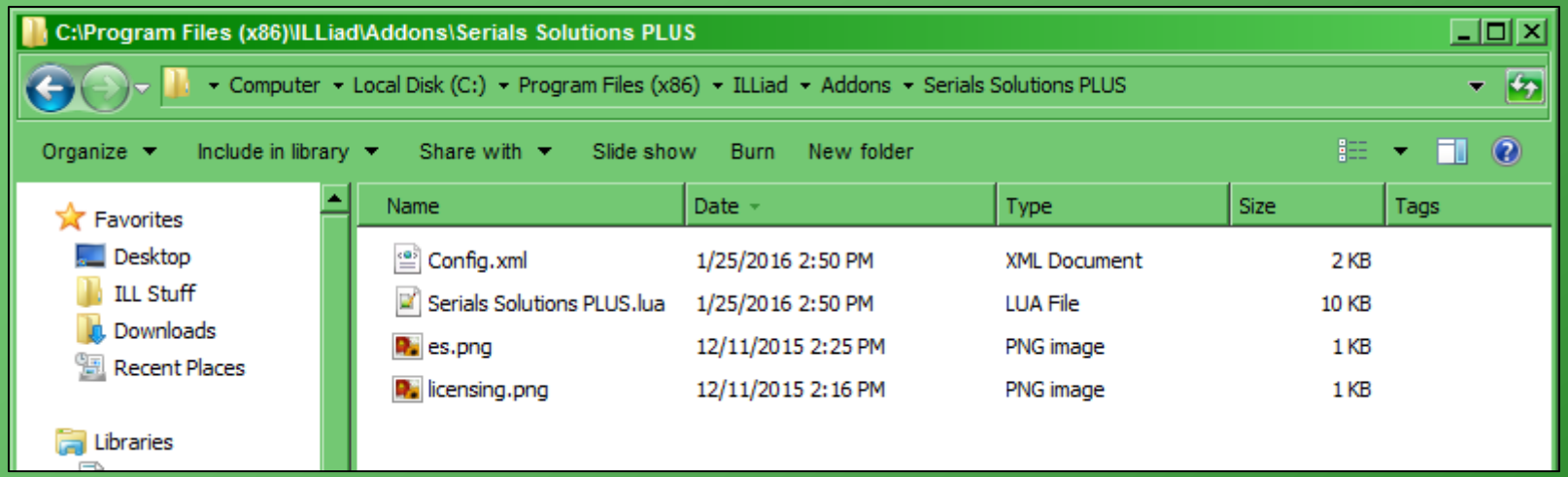
Getting Started

Try not to break anything.

- If your site uses a shared addons directory, you should always test on your computer first.
- The file path for addons is C:\Program Files\ILLiad\Addons. Make a shortcut to it.
  - Program Files (x86) for 64-bit operating systems after Vista



Every client addon is composed of at least two files:



- The Lua file contains the code that is executed when the transaction is opened.
- The XML file contains the settings that can be adjusted in the Manage Addons section of the client.
- Image and layout files may also be in this folder.

Basic Structure

Client addons all (mostly) share the same basic structure:

```
-- Addon version and revision notes (optional, but helpful), commented out.

local settings = {};
settings.Setting1 = GetSetting("Setting1");
settings.Setting2 = GetSetting("Setting2");

local interfaceMngr = nil;
local whateverForm = {};

whateverForm.Form = nil;
whateverForm.Browser = nil;
whateverForm.RibbonPage = nil;

require "Atlas.AtlasHelpers"; -- And any other libraries or files you need.

function Init() -- All addons need an Init function.

    interfaceMngr = GetInterfaceManager(); -- Gets the Interface Manager; vital for creating the addon interface.

    whateverForm.Form = interfaceMngr:CreateForm("Whatever", "Script"); -- Creates the addon form. This is essentially everything that appears within the addon tab.

    whateverForm.Browser = whateverForm.Form:CreateBrowser("Whatever", "Whatever Browser", "Whatever Ribbon"); -- Creates the browser for your addon, so you can navigate to web pages.

    whateverForm.Browser.TextVisible = false; -- Hides the annoying label for the browser. Trust me, it takes up a lot of space. You always want this hidden.

    whateverForm.Browser.WebBrowser.ScriptErrorsSuppressed = true; -- Suppresses Javascript errors on web pages. Always best to have this on, especially if you're dealing with WorldCat.

    whateverForm.RibbonPage = whateverForm.Form:GetRibbonPage("Whatever Ribbon"); -- Creates the ribbon above the tab; this is where your buttons go.

    whateverForm.RibbonPage:CreateButton("Pop-up!", GetClientImage("Wizard32"), "PopupFunction", "Button Space"); -- This is a button with the text "Pop-up!" It triggers the PopupFunction function, and is located in the Button Space group.

    whateverForm.Form:Show(); -- Makes the form visible. Usually you want that.


    MainFunction(); -- Runs MainFunction, which is defined below.

end

function MainFunction()

    whateverForm.Browser:Navigate("http://www.atlas-sys.com"); -- Navigates to the Atlas website.

end

function PopupFunction() -- Runs when the button is pressed;

    interfaceMngr:ShowMessage("Pop-up message: You pressed the button. Congratulations.", "Pop-up title: DING!");

end
```

Basic Structure - Settings

Settings are typically declared in the first lines of the addon. These are special variables that can be changed by users from the Manage Addons section of the client.

```
3   local settings = {};
4   settings.Setting1 = GetSetting("Setting1");
5   settings.Setting2 = GetSetting("Setting2");
6
```

Basic Structure – Form(s)

The InterfaceManager and any forms your addon uses are initialized next. The form includes everything in the tab created by the addon, whether it's visible or not.

```
7   local interfaceMngr = nil;
8   local whateverForm = {};
9
10  whateverForm.Form = nil;
11  whateverForm.Browser = nil;
12  whateverForm.RibbonPage = nil;
13
```

Basic Structure – Other Files (optional)

You may use the command "require" to allow the addon to access information in other files, such as the AtlasHelpers library.

```
13
14      require "Atlas.AtlasHelpers";
15
```

Basic Structure – Init function

This is the function that runs when the transaction is opened.

```
16  function Init()
17
18      interfaceMngr = GetInterfaceManager();
19
20      whateverForm.Form = interfaceMngr:CreateForm("Whatever", "Script");
21
22      whateverForm.Browser = whateverForm.Form:CreateBrowser("Whatever", "Whatever Browser", "Whatever Ribbon");
23
24      whateverForm.Browser.TextVisible = false;
25
26      whateverForm.Browser.WebBrowser.ScriptErrorsSuppressed = true;
27
28      whateverForm.RibbonPage = whateverForm.Form:GetRibbonPage("Whatever Ribbon");
29
30      whateverForm.RibbonPage:CreateButton("Pop-up!", GetClientImage("Wizard32"), "PopupFunction", "Button Space");
31
32      whateverForm.Form:Show();
33
34
35      MainFunction();
36
37  end
```

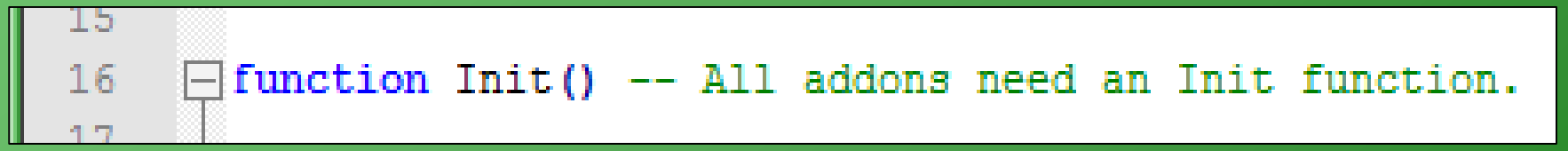
Basic Structure – Other Functions

Just because you *can* cram everything into the Init function doesn't mean you *should*.

```
39  function MainFunction()
40
41      whateverForm.Browser:Navigate("http://www.atlas-sys.com");
42
43  end
44
45  function PopupFunction()
46
47      interfaceMngr:ShowMessage("Pop-up message: You pressed the button. Congratulations.", "Pop-up title: DING!");
48
49  end
```

Basic Structure – Comments

Two hyphens denote a comment. Comments are not executed, but are helpful for others who may read your code.

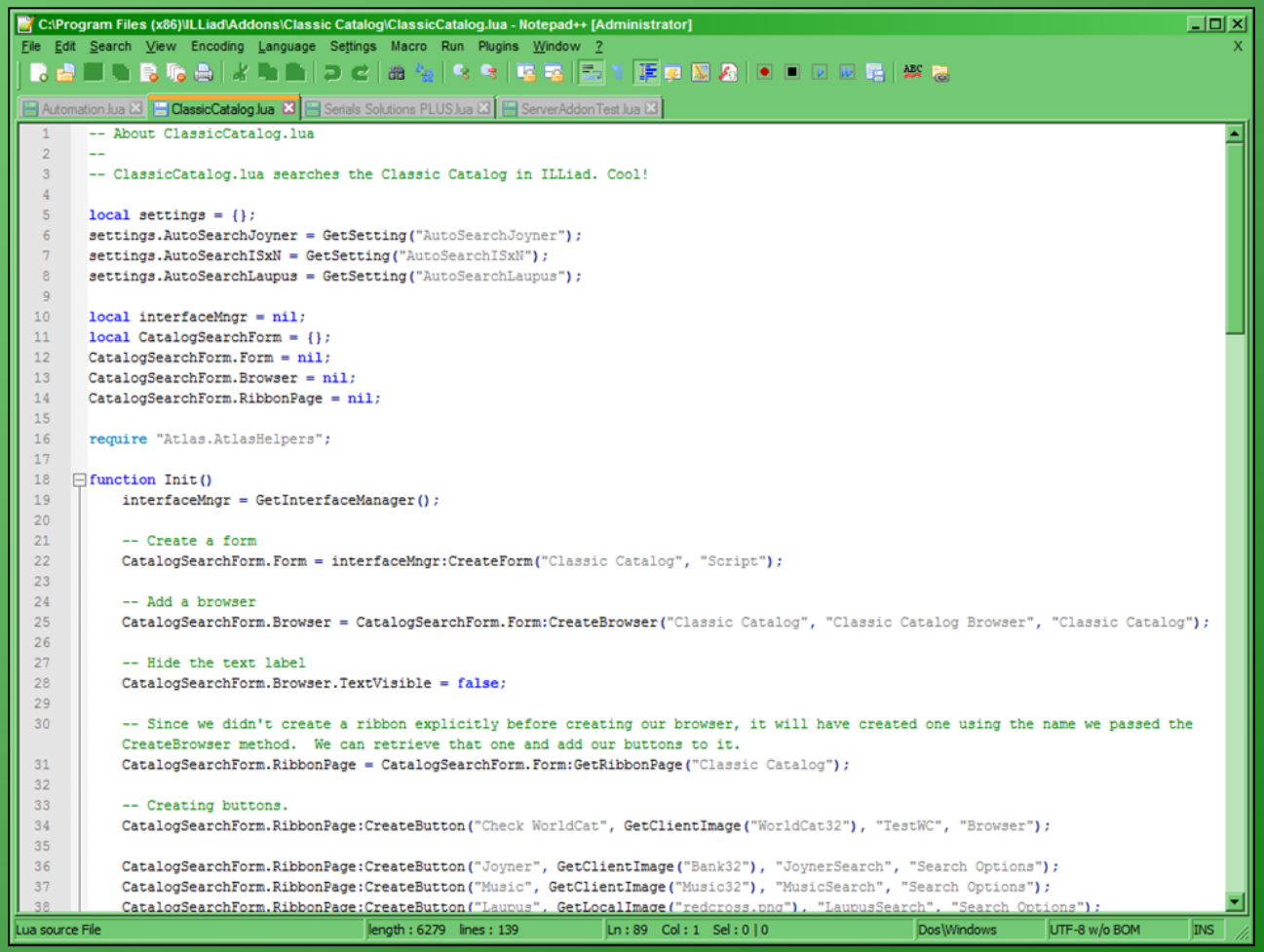


Basic Structure – Demo

Here's the result of the code in the previous slides:

Basic Structure – Easy Template

If this structure doesn't seem so basic, don't worry. You can always use existing addons as templates.



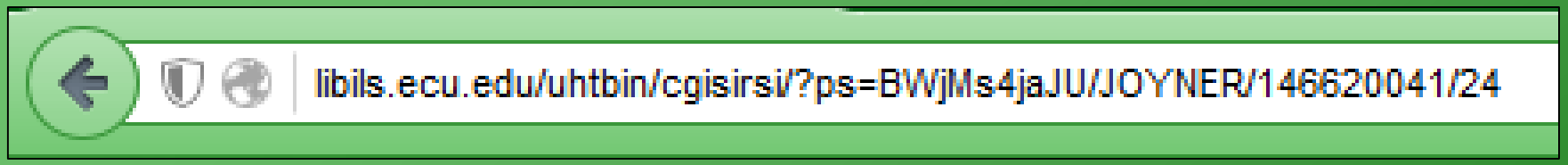
If-Then Statements

If-then statements, particularly in conjunction with GetFieldValue, are incredibly useful.

```
32  function Init()
33      local reqType = GetFieldValue("Transaction", "RequestType");
34      local process = GetFieldValue("Transaction", "ProcessType");
35      local queue = GetFieldValue("Transaction", "TransactionStatus");
36      local nvtgc = GetFieldValue("User", "NVTGC");
37
38      interfaceMngr = GetInterfaceManager();
39
40      if process == "Lending" then -- Ensures that these functions only run on Lending requests.
41          if settings.AVRenew and queue == "Awaiting Lending Request Processing" and reqType == "Loan" then
            --AVRenew only runs on loans in Awaiting Lending Request Processing.
42
43              AVRenew();
44
45          end
46
47          if settings.FigureCost then
48              FigureCost();
49          end
50
51          if settings.CancelClear and queue == "Awaiting Conditional Request Processing" then
52              SetFieldValue("Transaction", "ReasonForCancellation", nil);
53          end
54
55      end
```

Page Handlers

So, you want to make an addon for your OPAC, but your OPAC doesn't use nice, easily manipulated URLs?



That's where page handlers come in handy. That’s a link. Click it to read the page handlers documentation.

Page Handlers – Pick One

The type of page handler you use will depend on the page. When in doubt, there's always the custom page handler.

```
    if not banner:match("8%d%d%d%d%d%d%d%d") then
        interfaceMngr:ShowMessage("Improperly formatted or missing Banner ID.", "Patron needs Banner ID!");
        clipboard.SetData("Text", GetFieldValue("User", "LastName") .. ", " .. GetFieldValue("User", "FirstName"));
    else
        StatusForm.Browser:RegisterPageHandler("custom", "StatusCheck", "PatronStatus3", false);
        StatusForm.Browser:Navigate("http://libils.ecu.edu:8080/symws/rest/patron/lookupPatronInfo?clientID=ECULib_
    end

end

function PatronStatus3()
    local docText = StatusForm.Browser.WebBrowser.DocumentText;
    local status = docText:match("<statusType>.+</statusType>"):gsub("</*statusType>", "");

    if status ~= "OK" then
        interfaceMngr:ShowMessage("Patron is " .. status .. ".", "This patron is not okay!");
    end
end

function LoginCheck()
    local docText = StatusForm.Browser.WebBrowser.DocumentText;

    if docText:match("LoginUserResponse") then
        return true;
    else
        return false;
    end
end

function StatusCheck()
    local docText = StatusForm.Browser.WebBrowser.DocumentText;

    if docText:match("LookupPatronInfoResponse") then
        return true;
    else
        return false;
    end
end
```

Page Handlers – Dig Into HTML

Page handlers are pretty straightfoward, but you'll have to take a look at the HTML of the target page first.

Pattern Matching

Pattern matching is useful for creating custom page handlers as well as functions for automatically correcting or formatting data.

```
66  function Class.ISxNFix(str) -- This makes sure the ISxN is formatted correctly. Removes hyphens from
    ISBNs and cuts off extra characters, cuts off additional ISSNs, and formats ISSNs with a hyphen if they
    don't already have one.
67
68      str = str:gsub("-", "");
69
70      if str:match("%d%d%d%d%d%d%d%d%d%w+") then
71          str = str:match("%d%d%d%d%d%d%d%d%d%w+");
72      elseif str:match("%d%d%d%d%d%d%d%w") then
73          str = str:match("%d%d%d%d%d%d%d%w");
74          str = str:sub(1, 4) .. "-" .. str:sub(5, 8);
75      end
76
77      return str;
78  end
```

Config Files

Config files are relatively simple. These are where you set the version number and the settings for the addon.

```
3     <Name>Presentation Addon</Name>
4     <Author>Andrew Morgan</Author>
5     <Version>1.0.01</Version>
6     <Active>False</Active>
7     <Type>Addon</Type>
8     <Description>An example addon for my ILLiad Conference 2016 presentation.</Description>
9     <Forms>
10      <Form>FormRequest</Form>
11    </Forms>
12    <Settings>
13      <Setting name="Setting1" value="true" type="boolean">
14        <Description>Setting example 1.</Description>
15      </Setting>
16      <Setting name="Setting2" value="http://www.google.com" type="string">
17        <Description>Setting example 2.</Description>
18      </Setting>
19    </Settings>
20    <Files>
21      <File>Presentation Addon.lua</File>
22    </Files>
23  </Configuration>
```

Questions?

Feel free to e-mail me at morgana@ecu.edu.