Quick development of a mobile CICS application using Lua

Ezriel Gross
Circle Software

David Crayford
Fundi Software

Wednesday 6 August 2014
Session 15892

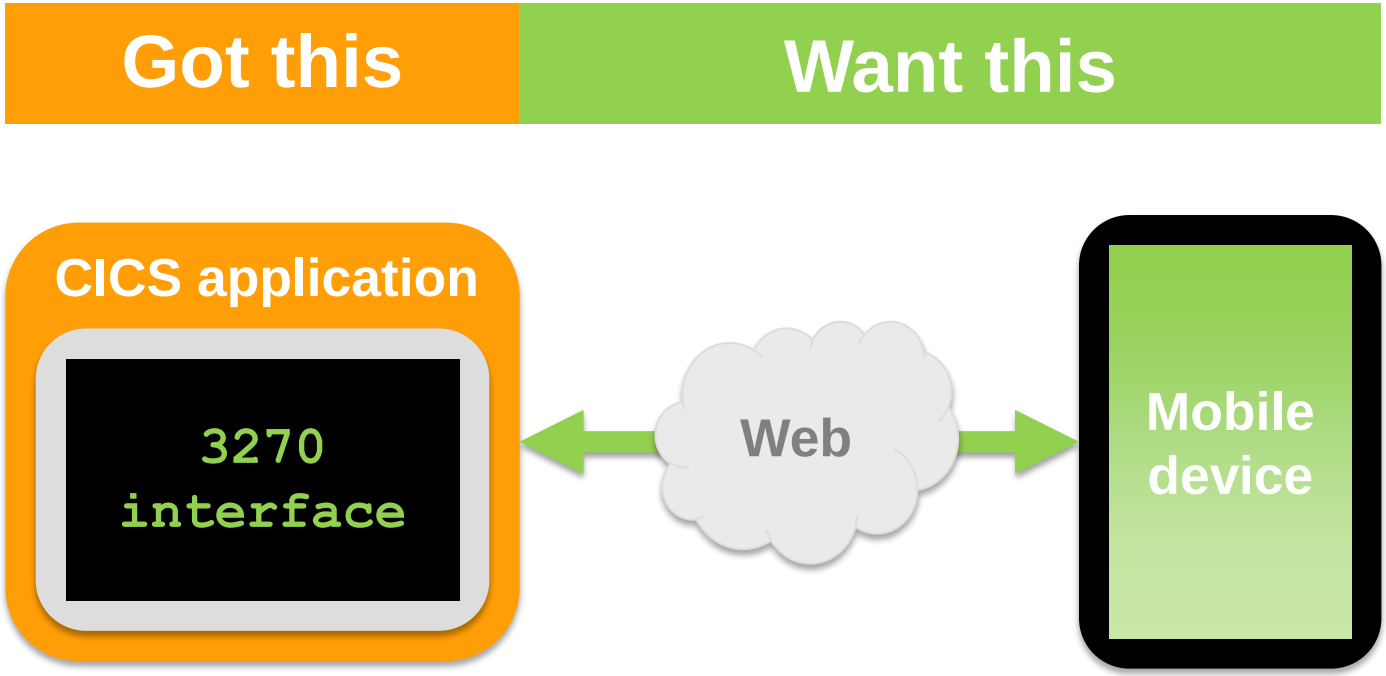
#SHAREorg

Agenda

- Got a CICS 3270 app, want it mobile… ***now!***
- Some options
- What's Lua?
- Why Lua?
- Using Lua to mobilize a CICS 3270 app
- Other uses for Lua on z/OS
- Introducing Lua4z

The emphasis here is ***rapid development***

Want to mobilize an existing CICS 3270 app



The CICS 3270 application: box office ticketing

```
Ticketing                                                    FBOCCM02
                          FUNDI BOX OFFICE

    Select an Action Code :


        Available Actions : 1  Find
                            2  Book
                            3  Reserve
                            4  Cancel

 Enter - Select   PF3 - End   PF12 - Cancel
```

Not a fully fledged production application: developed in-house to generate CICS and DB2 performance log data, not to book box office tickets!

The CICS 3270 application: box office ticketing

```
Ticketing                                                       FBOCCM07
Func : BOOK              FUNDI BOX OFFICE

       Complete the details below, then press ENTER

             Company Id :
                   Date :
               Venue Id :
                   Tier :
         Seats Requested :


                Seat(s) :
                  Aisle :
                    Row :
                  Block :




 Enter - Select   PF3 - End   PF12 - Cancel
```

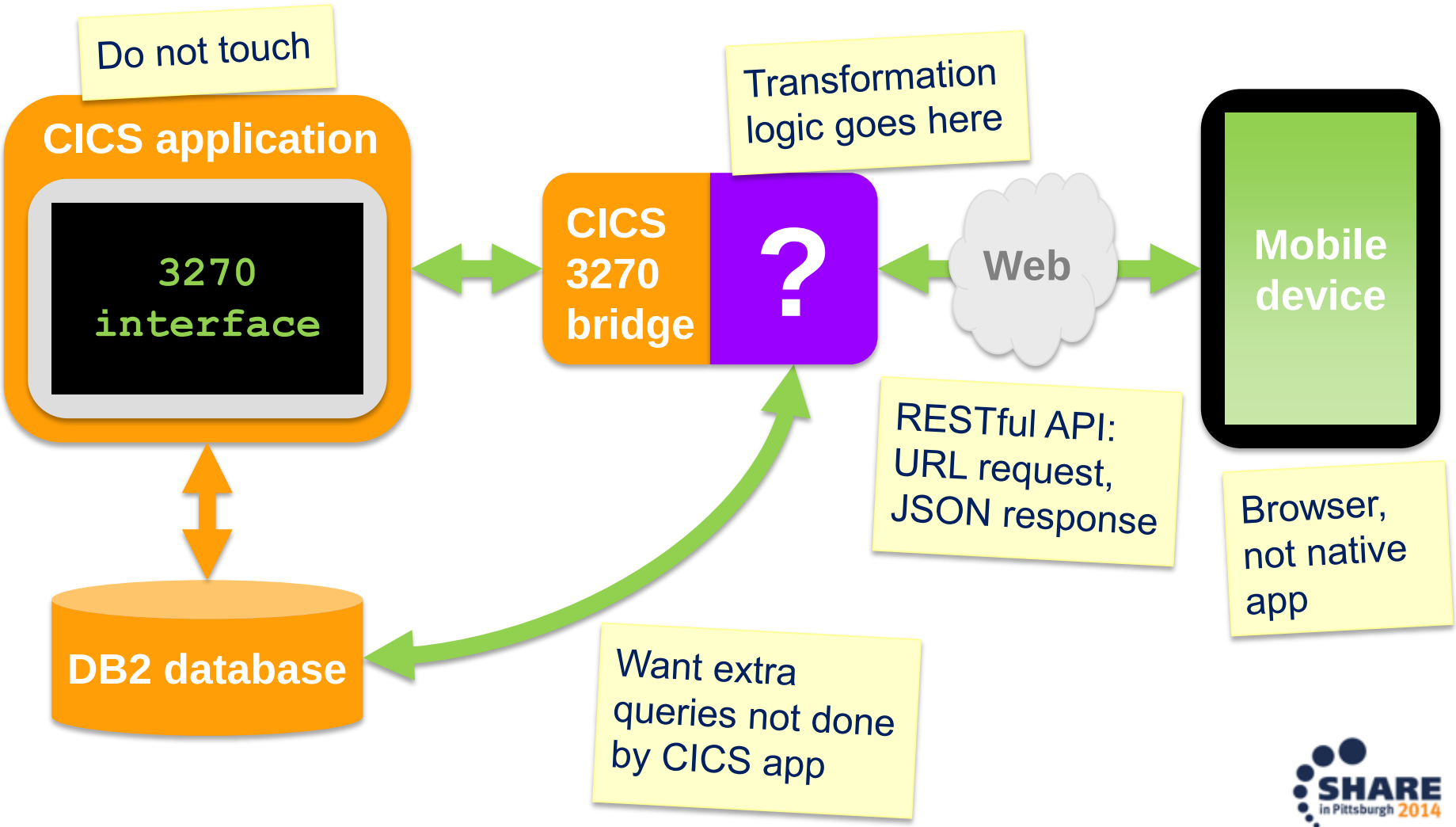
The CICS 3270 application: box office ticketing

```
Ticketing                                                       FBOCCM07
Func : BOOK              FUNDI BOX OFFICE

       Complete the details below, then press ENTER

             Company Id : EZTIK
                   Date : 2014-08-14
               Venue Id : 00263
                   Tier : G
         Seats Requested : 02


                Seat(s) : 10
                  Aisle : 31
                    Row : 02
                  Block : 04


 OK, seat(s) booked

 Enter - Select   PF3 - End   PF12 - Cancel
```

This application lacks list functions; for example, to list venues and event dates.

We need to add list functions to our mobile interface.

Details


Do not touch
CICS application
3270 interface
DB2 database
CICS 3270 bridge
?
Transformation logic goes here
Web
Mobile device
RESTful API: URL request, JSON response
Browser, not native app
Want extra queries not done by CICS app

Transforming between 3270 and JSON

COBOL copybook

```
01  M07DETI.
    02  FILLER PIC X(12).
    02  M07FUNCL     COMP  PIC  S9(4).
    02  M07FUNCF     PICTURE X.
    02  FILLER REDEFINES M07FUNCF.
      03 M07FUNCA     PICTURE X.
    02  FILLER   PICTURE X(2).
    02  M07FUNCI  PIC X(7).
    02  M07COIDL     COMP  PIC  S9(4).
    02  M07COIDF     PICTURE X.
    02  FILLER REDEFINES M07COIDF.
      03 M07COIDA     PICTURE X.
    02  FILLER   PICTURE X(2).
    02  M07COIDI  PIC X(05).
    02  M07DATEL     COMP  PIC  S9(4).
    02  M07DATEF     PICTURE X.
    02  FILLER REDEFINES M07DATEF.
      03 M07DATEA     PICTURE X.
    02  FILLER   PICTURE X(2).
    02  M07DATEI  PIC X(10).
    02  M07VUIDL     COMP  PIC  S9(4).
    02  M07VUIDF     PICTURE X.
    02  FILLER REDEFINES M07VUIDF. …
```

JSON

```
{
  "seats_requested" : 2,
  "venueID" : 263,
  "tier" : "G",
  "date" : "2014-07-01",
  "companyID" : "EZTIK"
}
```

Transformation logic

Which programming language do we write this in?

Some programming language options

- **COBOL** (or any “traditional” CICS programming language, such as C, PL/I, or assembler)
- **Java**
- **REXX**
- **PHP**
  - CICS TS Feature Pack for Dynamic Scripting V2.0
  - PHP scripts interpreted by Java runtime

and now…

- **Lua**

What's Lua?

- Powerful, fast, lightweight, embeddable scripting language
- Proven and robust
- Open source
- Many useful extensions
- Portable
  - Fundi has ported Lua, and various extensions, to z/OS
  - We've developed patches and extensions for z/OS
- Lua website: www.lua.org

"Lua" is Portuguese for "moon"

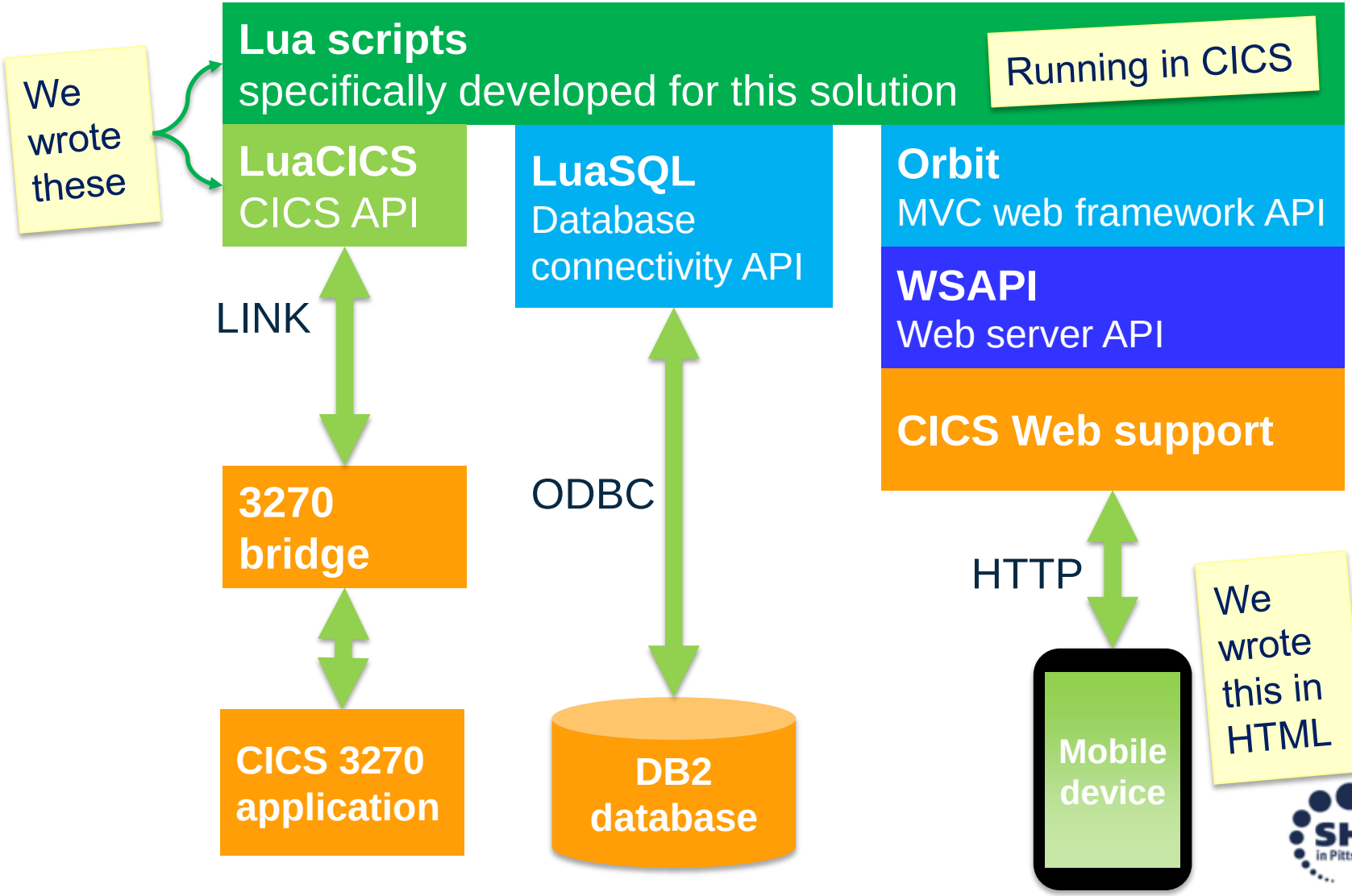
WIKIPEDIA
The Free Encyclopedia

Wikipedia uses Lua as a page templating engine

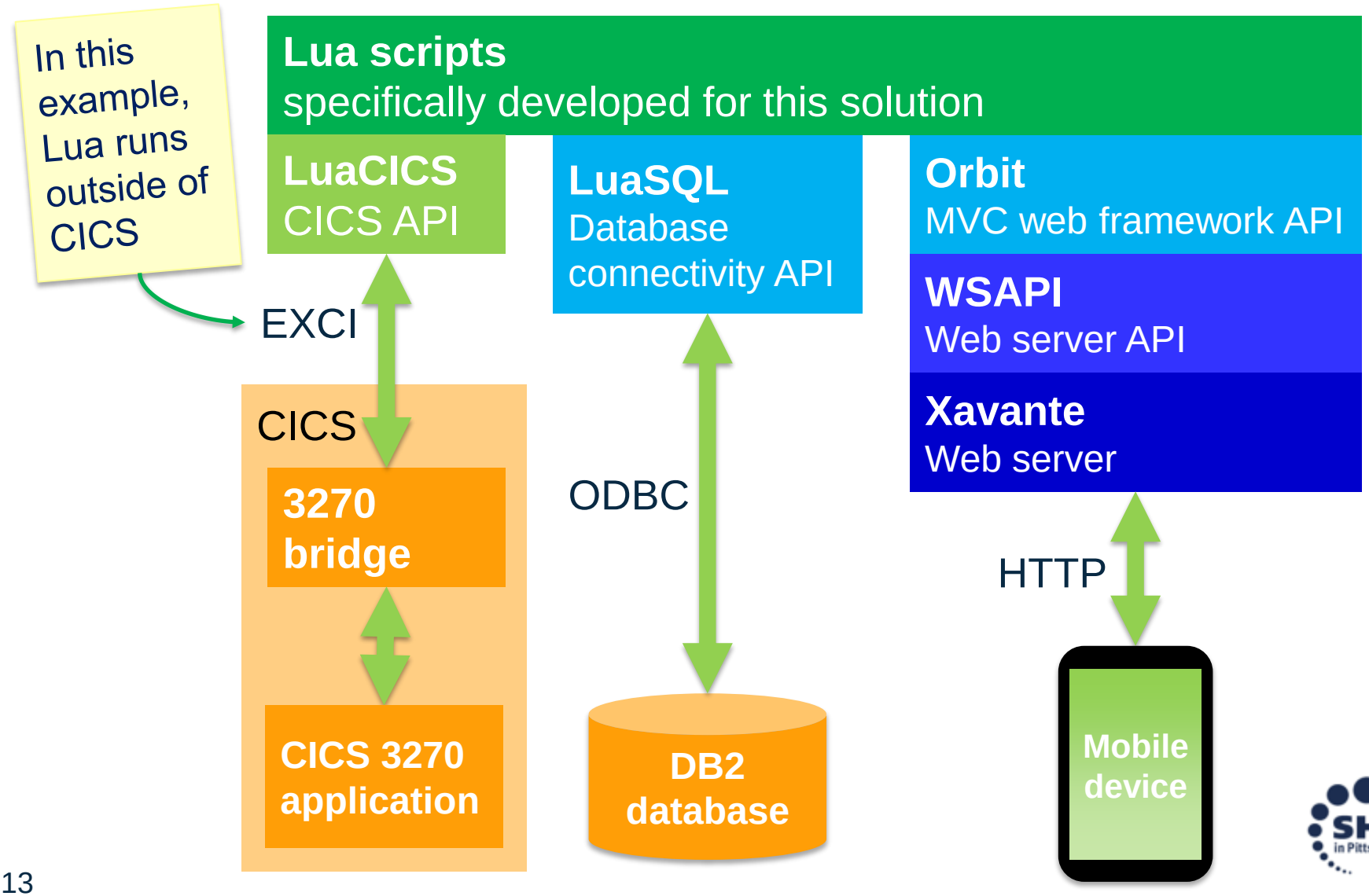
Why Lua?

- Fast execution speed
- Easy to learn, concise syntax
  - Especially if you already know JavaScript
- Many useful open-source extensions:
  - Multi-threaded web application stack
  - Support for XML and JSON
- Easy to extend
  - We developed new Lua functions that call the CICS Link3270 bridge
- Easy to embed in your existing application

Architecture of a Lua-based solution



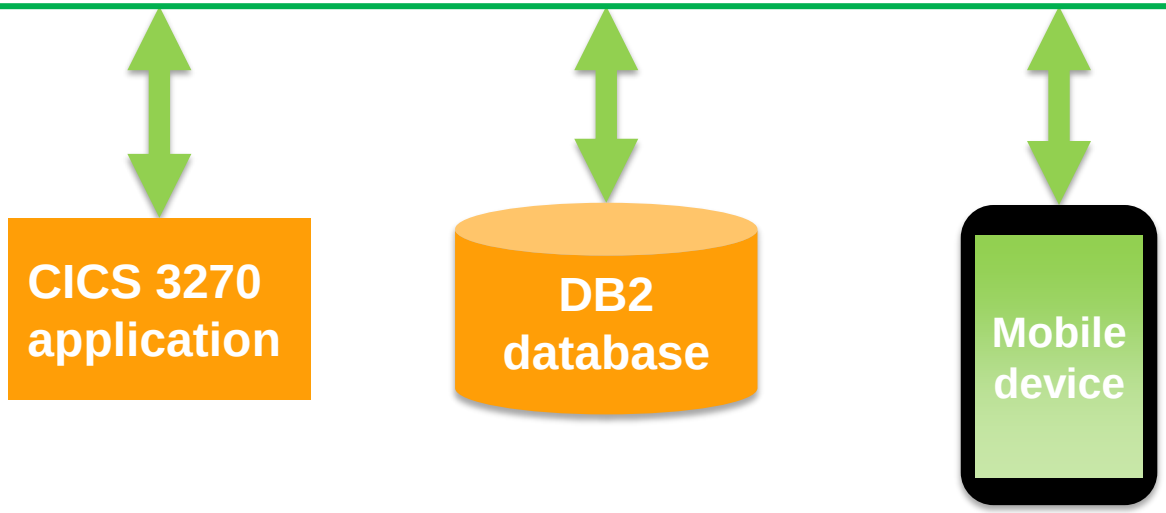
Our rapid development environment



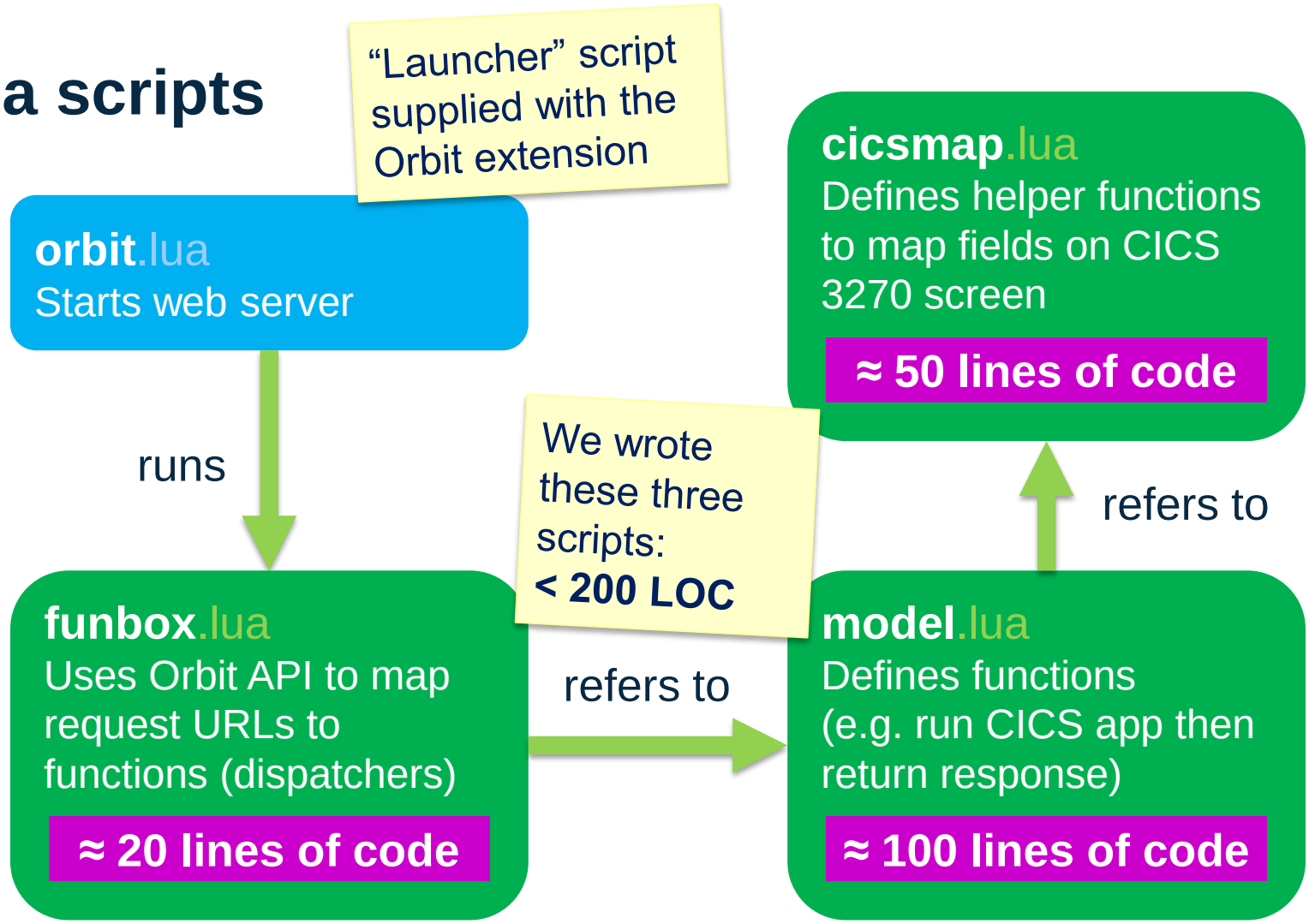
What do the Lua scripts do?

Lua scripts

- Map request URL to an action
- Interact with the CICS 3270 application
- Query DB2
- Transform: JSON to 3270, JSON to SQL, SQL results to JSON
- Return JSON

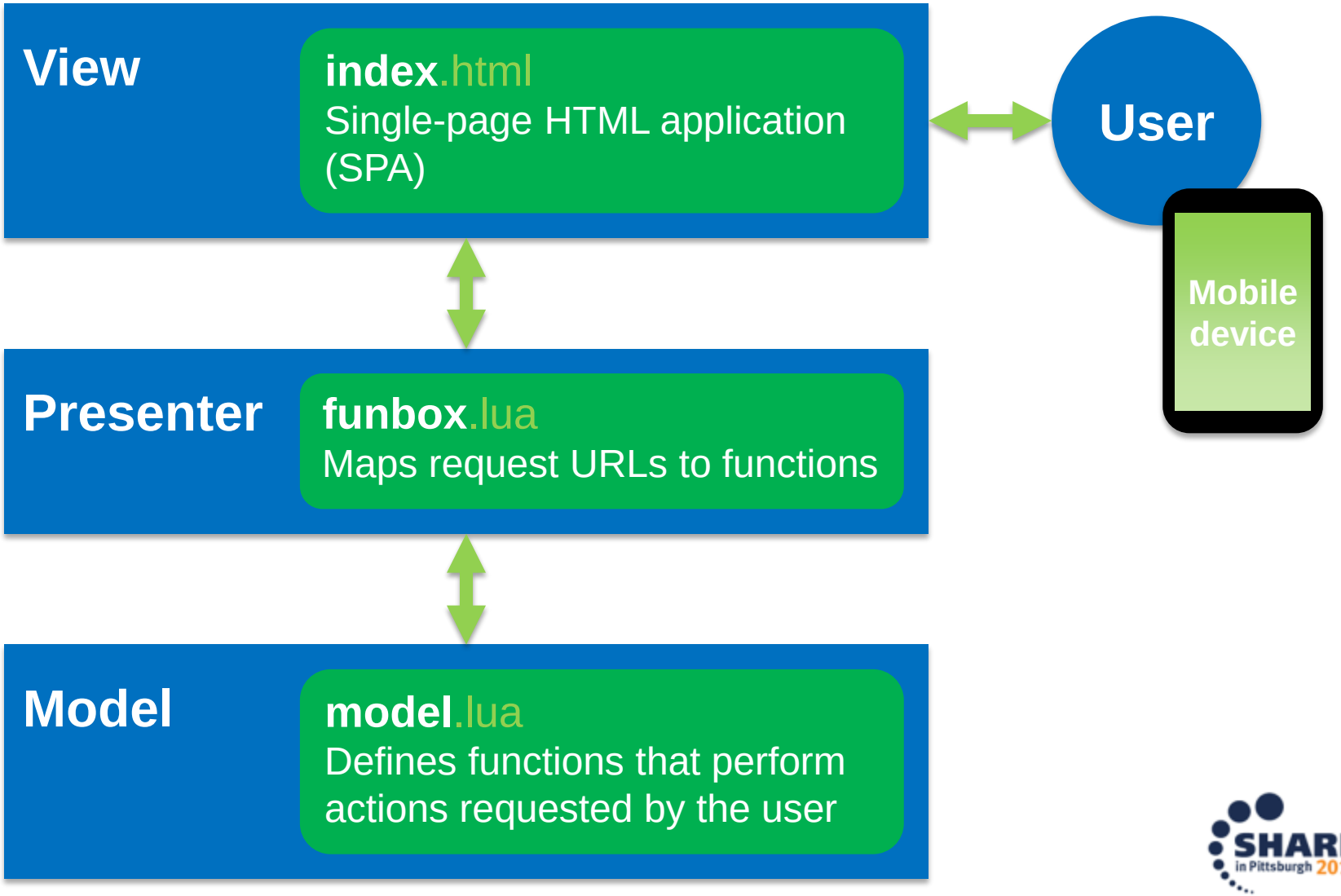


Lua scripts



These are MVS PDS members. We could have stored these Lua scripts as zFS files with a **.lua** file extension.

Model-view-presenter (MVP) design pattern



Mapping request URLs to functions

funbox.lua

RESTful API request URL	**HTTP method**	**Function** (defined in model.lua)
/funbox/events	Get	`list_events(…)`
/funbox/venues	Get	`list_venues(…)`
/funbox/venues/*venueid*	Get	`list_venues(…,`*`venueid`*`)`
/funbox/book?*details*	Post	`book_event(…)`

Requested ticket details sent as query string; we could have sent JSON in the request body

This function uses the CICS 3270 application; the others query DB2

Mapping request URLs to functions

funbox.lua

```
local orbit = require("orbit")
local Model = require("model")
local R = require("orbit.routes")
... a few other required modules ...
local funbox = orbit.new()
local model = Model.new()
-- Dispatcher for static HTML file
funbox:dispatch_static("/index%.html")
-- Dispatcher for venues
funbox:dispatch_get(
  function (web, parms) return model:list_venues(web, parms.id) end,
  R "/funbox/venues/:id")

other dispatchers…
```

'FUNBOX.LUA(MODEL)' contains our functions

Call this **function** when the client requests this **URL** (:id is a variable value)

Submitting SQL queries, returning JSON

model.lua

… preamble code and helper functions shown on next slide …

```
function model:list_venues(web, id)
  local query = QueryBuilder.new()
  query:select("*"):from("funbox.venue")
  if id then
    query:where("venueid =", id)
    return get_response(web, get_query(self, query))
  end
  return get_response(web, list_query(self, query))
end
```

Create an SQL query

Add WHERE clause for specific venue

Alternative code path handles a request for all venues (no venue ID in URL)

```
local class = require "middleclass"
local luasql = require "luasql.odbc"
local json = require "cjson"
local QueryBuilder = require("query")
local model = class("funbox.model") -- this class
-- Constructor
function model:initialize()
  local env = luasql.odbc()
  self.dbcon = assert( env:connect() )
end
-- Submits SQL query
local function get_query( self, query )
  query = tostring(query)
  local cur = assert( self.dbcon:execute( query ) )
  return cur:fetch( {}, "a" )
end
-- Returns the response from a GET request
local function get_response(web, res)
  if not res then
    web.status = 404
    return nil
  end
  web.headers["content-type"] = "application/json";
  return json.encode(res)
end
```

Return SQL results in Lua table

Return Lua table as JSON

Mapping CICS 3270 application data

cicsmap.lua

Packs/unpacks data to/from 3270 screen map. Possible candidate for code generation.

```
local struct = require "struct"

-- Format string
local fmtstr =
  "c24 H c3 c5 h c3 c10 h c3 c5 h c3 c h c3 c2 c47 h c3" ..
  "c2 c40 h c3 c2 c32 H c3 c2 c12 H c3 c2 c71 c64 c5 c64 c53"

-- Length of the structure
local LENGTH = 488

-- Helper function to unpack the structure
local function unpack(self, s)
  self.filler0, self.companyid_len, self.filler1, self.companyid,
  self.date_len, self.filler2, self.date, self.venueid_len, self.filler3,
  self.venueid, self.tier_len, self.filler4, self.tier, self.num_seats_len,
  self.filler5, self.num_seats, self.filler6, self.seats_length, self.filler7,
  self.seats, self.filler8, self.ailse_len, self.filler9, self.aisle,
  self.filler10, self.row_length, self.filler11, self.row, self.filler12,
  self.block_len, self.filler13, self.block, self.fillerbig, self.msg,
  self.filler14, self.msg2,self.filler15
  = struct.unpack(fmtstr, s)
end
```

continued…

continued from previous slide

```
-- Unpack method
function M:unpack(s)
  unpack(self, s)
end

-- Unpack method to pack the table to a string
function M:pack()
  return struct.pack(
    fmtstr,
    self.filler0, self.companyid_len, self.filler1, self.companyid,
    self.date_len, self.filler2, self.date, self.venueid_len, self.filler3,
    self.venueid, self.tier_len, self.filler4, self.tier, self.num_seats_len,
    self.filler5, self.num_seats, self.filler6, self.seats_length, self.filler7,
    self.seats, self.filler8, self.ailse_len, self.filler9, self.aisle,
    self.filler10, self.row_length, self.filler11, self.row, self.filler12,
    self.block_len, self.filler13, self.block, self.fillerbig, self.msg,
    self.filler14, self.msg2,self.filler15
    )
end

return M
```

Interacting with the CICS 3270 application

model.lua

```
local mapper = require "cicsmap"
local link3270 = require "cics.link3270x"

function model:book_event(web, parms)
  -- Decode the POST JSON payload
  local req = json.decode(web.POST.post_data)
  -- Open the Link3270 bridge session
  local br = link3270.open {
    applid = "LUATCIC1",
    wait_interval = 300,
  }
  -- Create the screen mapper
  local ads = mapper.new()
  ads.companyid = req.companyID;
  ads.companyid_len = #req.companyID;
  ads.venueid = string.rjust(tostring(req.venueID),5,"0")
  ads.venueid_len = #ads.venueid
  ads.date = req.date
  ads.date_len = #req.date
```

Insert the requested ticket details into the screen map variables

continued…

continued from previous slide

```
  -- Run the transaction with no input to simulate running from 3270
  br:exec("FBTB")

  -- Receive map, TRANSID(FBTB) MAP(M07DET) MAPSET(FBOMSET)
  local resp = br:receive_map("FBTB", "M07DET", "FBOMSET",
ads:pack())

  -- Check the result
  local sent_map = ads:unpack(resp)
  if sent_map.msg:sub(1,2) ~= "OK" then -- No "OK" message
    web.status = "400 Transaction Failed"
    return sent_map.msg
  end
end
```

Default response is status 200, OK

Complete listing of funbox.lua

```
local orbit = require("orbit")
local R = require("orbit.routes")
local Model = require("model")
local funbox = orbit.new()
local model = Model.new()

-- Dispatcher for static file
funbox:dispatch_static("/index%.html")

-- Dispatcher for venues
funbox:dispatch_get( function (web, parms)
  return model:list_venues(web, parms.id) end, R "/funbox/venues/:id")

-- Dispatchers for events
funbox:dispatch_get( function (web, parms)
  return model:list_events(web, parms) end, R "/funbox/events")

funbox:dispatch_post( function (web, parms)
  return model:book_event(web, parms) end,  R "/funbox/book")

return funbox
```

Running the solution from an MVS batch job

```
/ORBIT    JOB CLASS=W,MSGCLASS=T,NOTIFY=&SYSUID
//*
//LUA       EXEC PGM=LUA,PARMDD=SYSIN
//STEPLIB   DD   DISP=SHR,DSN=LUA.LOAD
//          DD   DISP=SHR,DSN=CICS.V690BASE.CICS.SDFHLOAD
//          DD   DISP=SHR,DSN=CICS.V690BASE.CICS.SDFHEXCI
//SYSPRINT  DD   SYSOUT=*
//DSNAOINI  DD   DISP=SHR,DSN=FUNBOX.CNTL(DSNAOINI)
//LUA       DD   DISP=SHR,DSN=FUNBOX.LUA
//HTML      DD   DISP=SHR,DSN=FUNBOX.HTML
//SYSIN     DD   *
posix(on) / orbit -p 7030 //lua(funbox)
/*
```

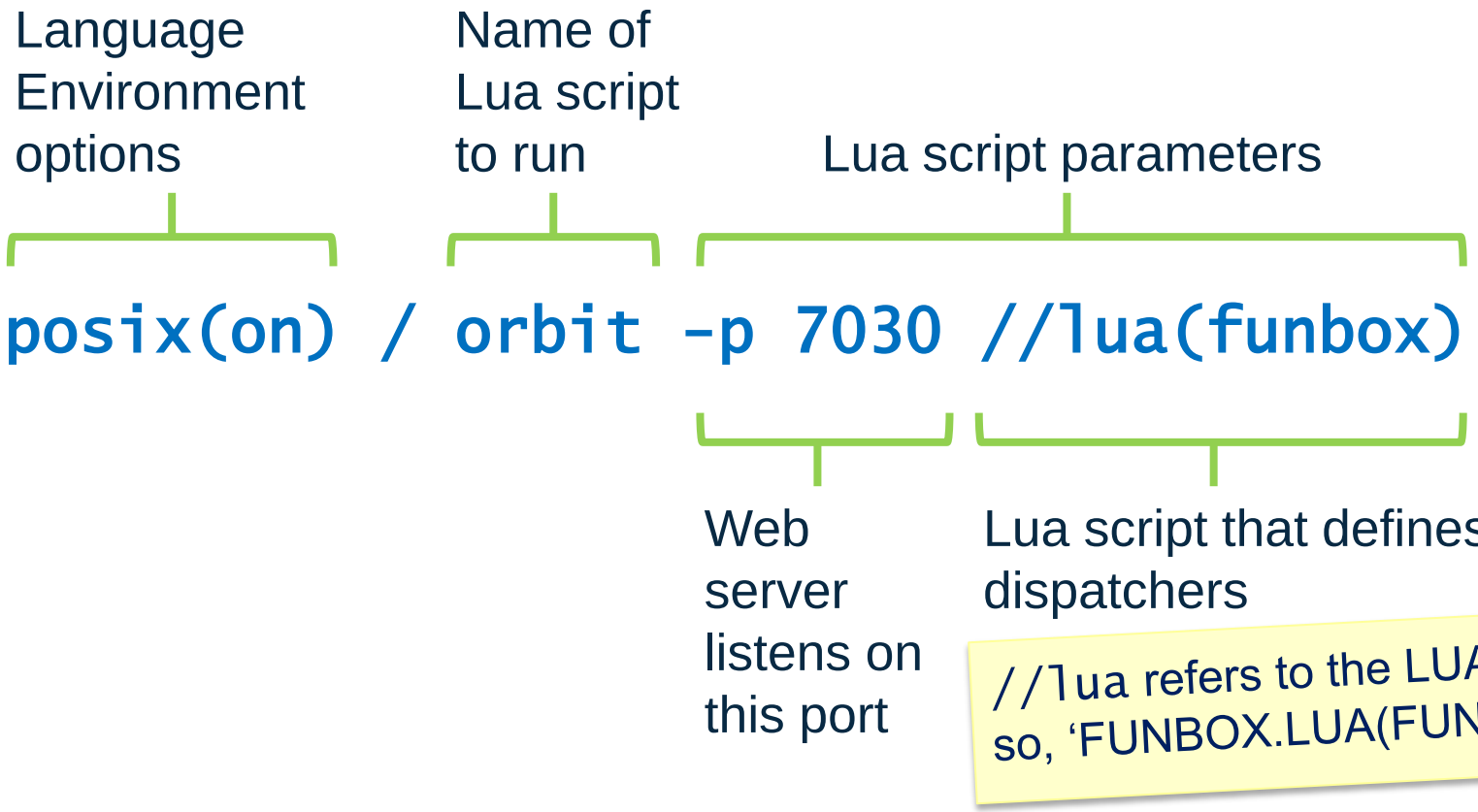
Lua interpreter

Need CICS libraries for EXCI

Text file containing ODBC connection details

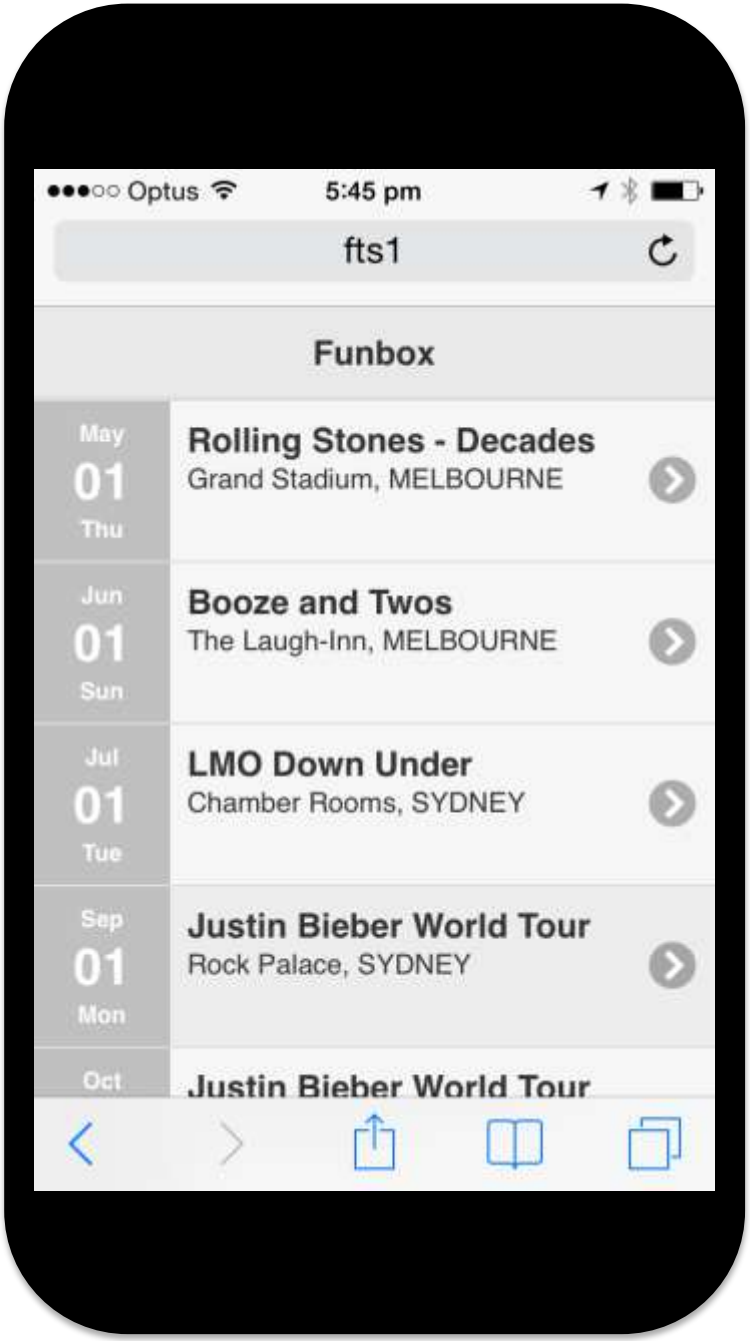
Could also run from TSO foreground or z/OS UNIX shell

SYSIN: input to the Lua interpreter



Mobile web interface: Listing events

1. User opens index.html (our single-page app, or **SPA**) in their web browser
2. SPA uses jQuery API to send Ajax get request for the URL **funbox/events**
3. Lua function **list_events()** submits SQL query, returns list of events in JSON
4. SPA presents list of events using jQuery Mobile Listview widget
5. User selects an event…



jQuery Ajax get example

Displays the JSON data in a jQuery Mobile Listview widget

index.html

```
getJSON("funbox/events").done(showEvents);
…
function getJSON(url) {
  return $.ajax({
    type : "GET",
    url : url,
    dataType : "json",
    beforeSend : function() {
      $.mobile.loading("show");
    }
  }).always(function() {
    $.mobile.loading("hide");
  }).fail(
      function(jqXHR, textStatus) {
        alert("Request failed: " + textStatus + " - "
            + jqXHR.responseText + "\nurl: " + this.url);
      });
}
```

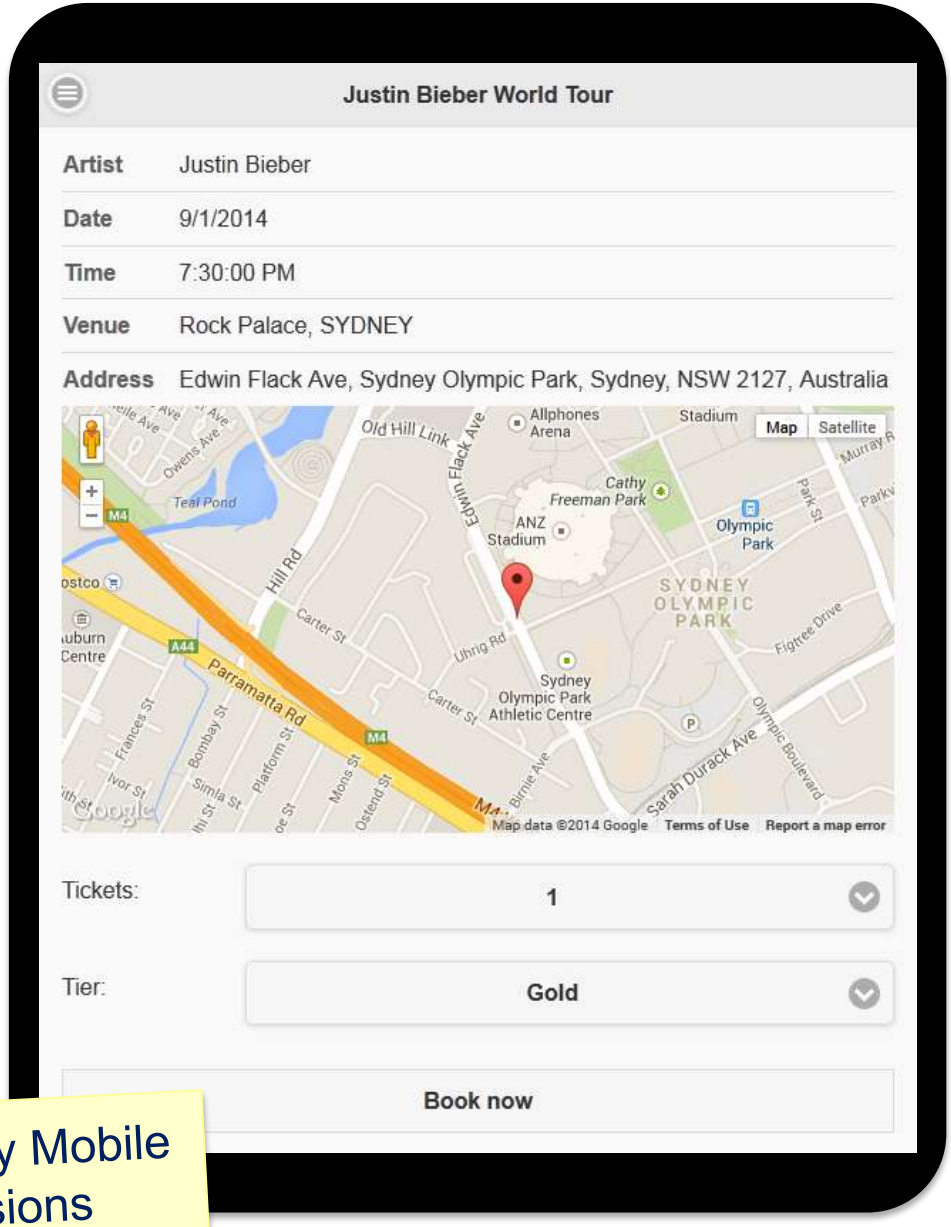
Gets

```
[{"TOURID":"RSDEC ",
"ARTIST":"Rolling Stones",
"TOURNAME":"Rolling Stones - Decades",
"TIME":"20:00:00",
"VENUENM":"Grand Stadium, MELBOURNE",
"STARTDATE":"2014-05-01",
"VENUEID":192}
… ]
```

Mobile device

Mobile web interface: Viewing event details

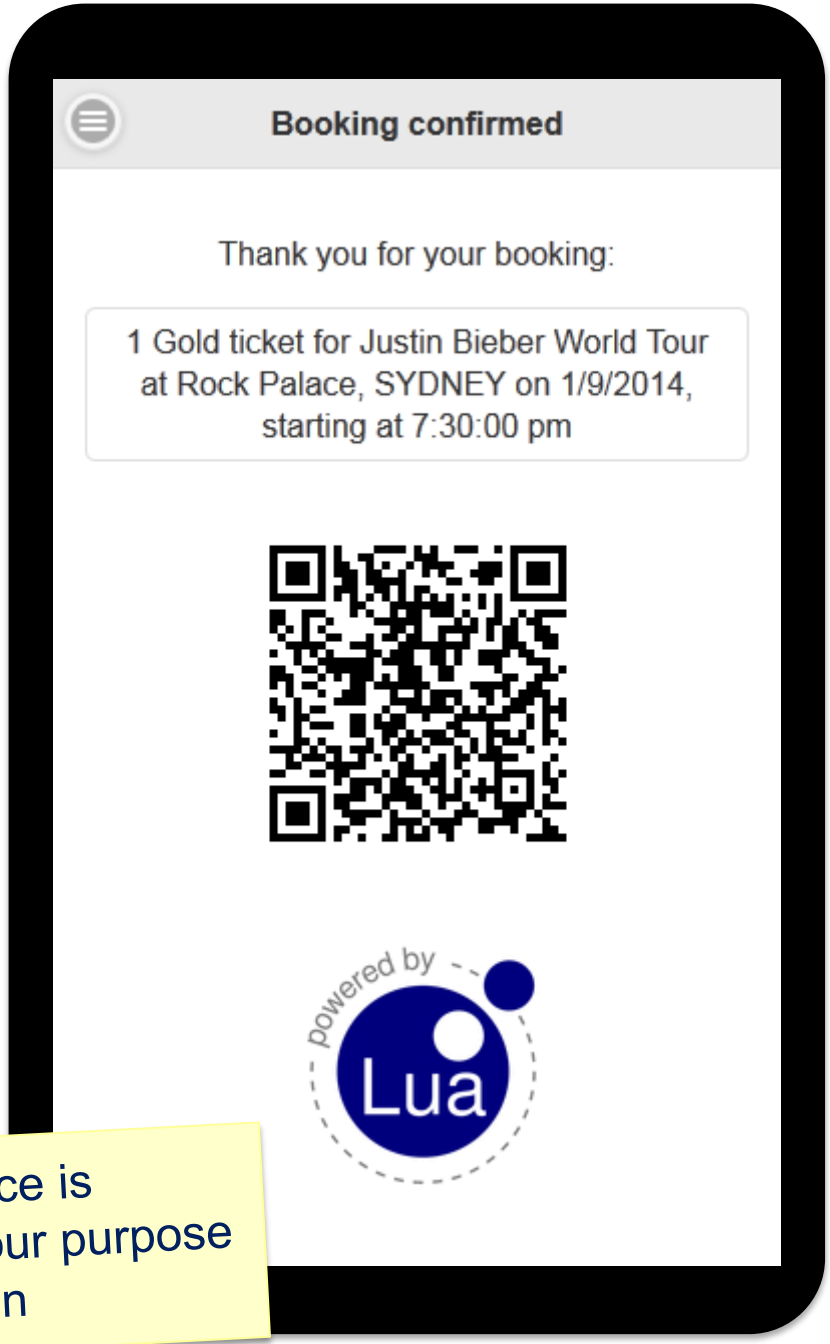
1. SPA sends Ajax get request for the event
2. Lua function returns event details as JSON
3. SPA presents event details, using Google Maps API to show venue
4. User selects number of tickets and tier, clicks **Book now**



Responsive web design (RWD): jQuery Mobile CSS adapts to different screen dimensions

Mobile web interface: Booking confirmation

1. SPA sends Ajax “Post” request with desired ticket details
2. Lua script interacts with CICS application
3. If the response from the CICS application indicates success, the Lua script returns a “success” (OK) response
4. SPA displays confirmation message



Yes, this user interface is simplistic, but it fits our purpose for this demonstration

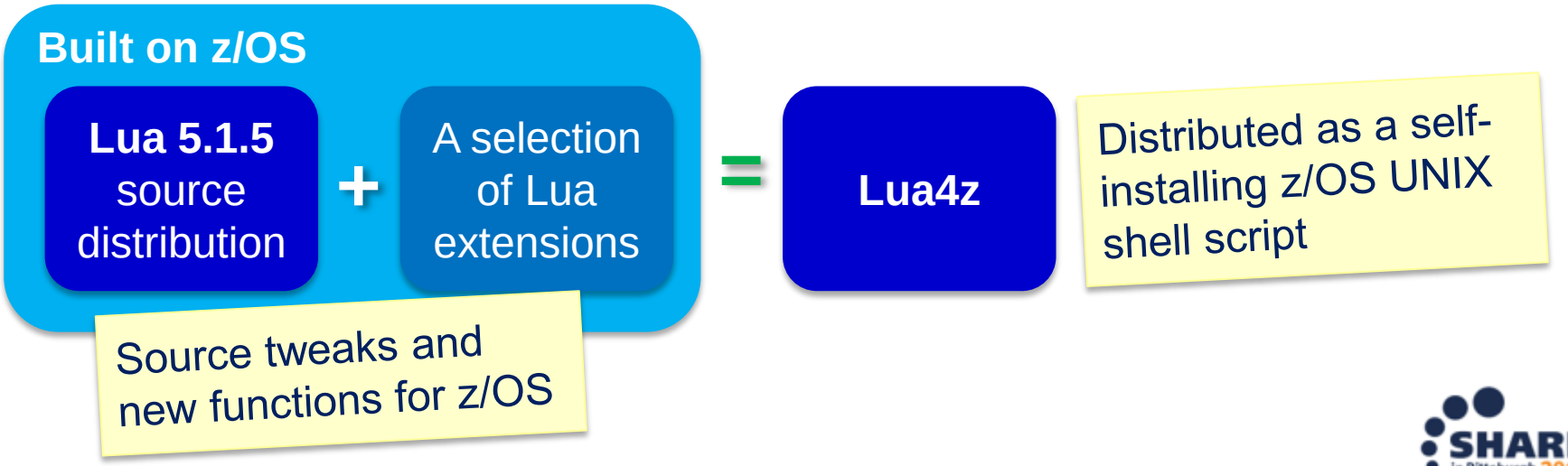
Other uses for Lua on z/OS

- Alternative to REXX:
  - Much faster execution
  - Concise syntax
  - Built-in (standard library) support for regular expressions
  - Thread support
  - Many useful open-source extensions, including web frameworks, XML parsing, JSON… and many, many more: see the list on the Lua users wiki at lua-users.org/wiki/LibrariesAndBindings
  - Modular source structure via `require()` function
  - Active user and development community
- Embedding in your own applications

Lua4z: a binary distribution of Lua for z/OS, with extensions

- Runs in MVS batch, TSO batch or foreground, z/OS UNIX
- Work in progress: **alpha release coming soon!**

- Initial plan:
  - Free version with no support contract
  - Support contracts
  - For-money extensions, such as LuaCICS



Lua extensions we might include with Lua4z

ansicolors	A simple Lua function for printing to the console in color
Bitfield	Alternative to the standard Lua bit32 library
busted	Unit testing framework
CGILua	Tool for creating dynamic web pages and manipulating input data from web forms
cliargs	Command-line argument parser for Lua
Copas	Coroutine Oriented Portable Asynchronous Services for Lua
Copas Timer	Socket scheduler
Cosmo	Safe templates engine
Coxpcall	Coroutine-safe xpcall and pcall versions
dkjson	A module for encoding and decoding JSON
inspect	Lua table visualizer, ideal for debugging
LDoc	A Lua documentation tool
lpack	Lua library for packing and unpacking binary data
Lpeg	Parsing Expression Grammars For Lua
Lua BitOp	Lua Bit Operations Module
Lua CJSON	Provides JSON support for Lua
Lua Haml	Implementation of the Haml markup language
Lua Lanes	Multithreading library
lua_zlib	Interface to zlib
LuaCICS	Lua4z extension for CICS
Lua-cmsgpack	MessagePack implementation and bindings for Lua
LuaCrypto	Lua frontend to the OpenSSL cryptographic library
LuaDate	Date and time module
LuaDBI	A database interface library
LuaExpat	SAX XML parser based on the Expat library
LuaFileSystem	File system library

This list is subject to change: we might add or remove some

continued…

Lua extensions we might include with Lua4z

Lua-iconv	POSIX iconv binding (performs character set conversions)
LuaISPF	Lua4z extension for ISPF services
LuaJSON	Customizable JSON decoder/encoder
LuaLogging	A simple API to use logging features
luaposix	Lua bindings for POSIX (including curses)
LuaSec	A binding for the OpenSLL library to provide TLS/SSL communication
LuaSocket	Network support
lua-Spore	Generic ReST client (SPORE: Specification to a POrtable Rest Environment)
LuaSQL	Database connectivity API
Luassert	Extends Lua's built-in assertions
Luaunit	A unit-testing framework
lua-websockets	Websockets: provides sync and async clients and servers for copas
Markdown	A pure-Lua implementation of the Markdown text-to-HTML markup system
Mercury	A Sinatra-like web framework (or DSL, if you like) for creating web applications in Lua
middleclass	A simple OOP library for Lua
MoonScript	A programmer friendly language that compiles to Lua
Orbit	An MVC framework for Lua
Orbiter	A self-contained personal web framework
Penlight	Lua utility libraries loosely based on the Python standard libraries
redis-lua	A Lua client library for the redis key value storage system
Rings	Create new Lua states from within Lua
say	String hashing/indexing library: useful for internationalization
struct	Library for converting data to and from C structs
Telescope	A test/spec library for Lua
VStruct	Functions for manipulating binary data
WSAPI	Web server API
Xavante	Web server

Contact

- Send email to:

  info@lua4z.com

- Ask questions about Lua4z

- Register your interest in becoming a Lua4z alpha tester

Please evaluate this session

Questions?