LassoWARE
Tools for Rapid Lasso Development

Standardize and Automate

Let Your Code Do the Work

Building RAD Systems in LDML

Greg Willits

Outline

Rapid Application Development

Step 1 - The Corral Method

Structure - Beyond the Method

Managers - Divide and Conquer

Modules - Plug and Play Corral

But, first . . .

Armadillo Facts

- 20 species - built to dig!
- Only the 3-banded Armadillo can roll up
- USA: Nine-Banded Armadillo
- Always has four identical young, all the same sex
- Are good swimmers! Can go underwater for 4-6 minutes
- Long sticky tongue, few stubby molars — insects, plants
- Ancient creature, warm blooded, but poor at it
- Used for immune research
- Jumps straight up when startled

http://www.msu.edu/~nixonjos/armadillo/facts.html

Rapid Application Development

Looking at your code
from a different perspective

Rapid Application Development

- It's a software design process
  - rapid ≠ quick and dirty!
- Objectives
  - get to executable UI / code ASAP
  - meet most requirements soon, vs. meet all requirements later
  - implement before requirements changes
- Drivers
  - life cycle, dynamic requirements

Traditional vs RAD

Traditional

- Low level language
- Text development tools
  BBEdit, MPW, CodeWarrior
- Optimized performance
- Redundant for perf.
- Limited reusability is OK
- Hard coded
- Get it “right”

RAD

- High level language
- Usually GUI dev tools
  GoLive, HyperCard, Project Builder
- Optimized development
- Minimal redundancy
- Wide reusability
- Abstracted
- Get it implemented

Elements/Benefits of RAD

- Reiterative prototyping involves users
  - repetitive design, test, review
- Reusable code (libraries, objects)
  - general purpose, parameter driven
- Reduced redundancy
  - no time for non-value-added maintenance
- Small teams of "Renaissance" authors
  - multi-skilled: analyze, design, program

Relative Sacrifices of RAD

- Features
  - limitated to high level tool capabilities
- Performance
  - abstraction adds layers
- Reliability (?)
  - collisions/gaps in layers/managers
  - reduced QC, testing

RAD and WWW

- Web by nature is RAD
- High level languages: html, ldml, javascript
- “Now” more important than “better”
- Reusability / dynamics over performance
- Renaissance artisans — are we not they!

RAD and Lasso

- High level language, GUI tools via Studio
- Abstraction vs. throughput optimization
  - ldml inline vs. -sql
  - everything can be a variable
  - Corral vs. linear
- Let's leverage language capabilities
  - create RAD components, tools, frameworks
  - automate the code to write itself
  - LDML 5/6 better equipped

Step 1: The Corral Method

“God bless Peter D. Bethke”

-- Chris Corwin

Corral Lexicon

- **Template:** HTML layout structure, defines placeholders for content (general purpose)
- **Page Blocks:** files with discrete content components or snippets (specific)
- **Stub file:** defines page blocks to use in template (communicates), includes siteConfig
- **SiteConfig:** defines variables/constants used by most pages, sets conditionals based on page location / name (initializes)

Loading a Page: Classic Corral

Stub

```
[incl:siteConfig]

[var:leftBlock=
   'menu.las']
[var:mainBlock=
   'special.las']

[incl:template]
```

PageBlocks

```
Please buy my
widgets. They
are the finest!

Summit Special!
Free Armadillo
with 2 widgets.
```

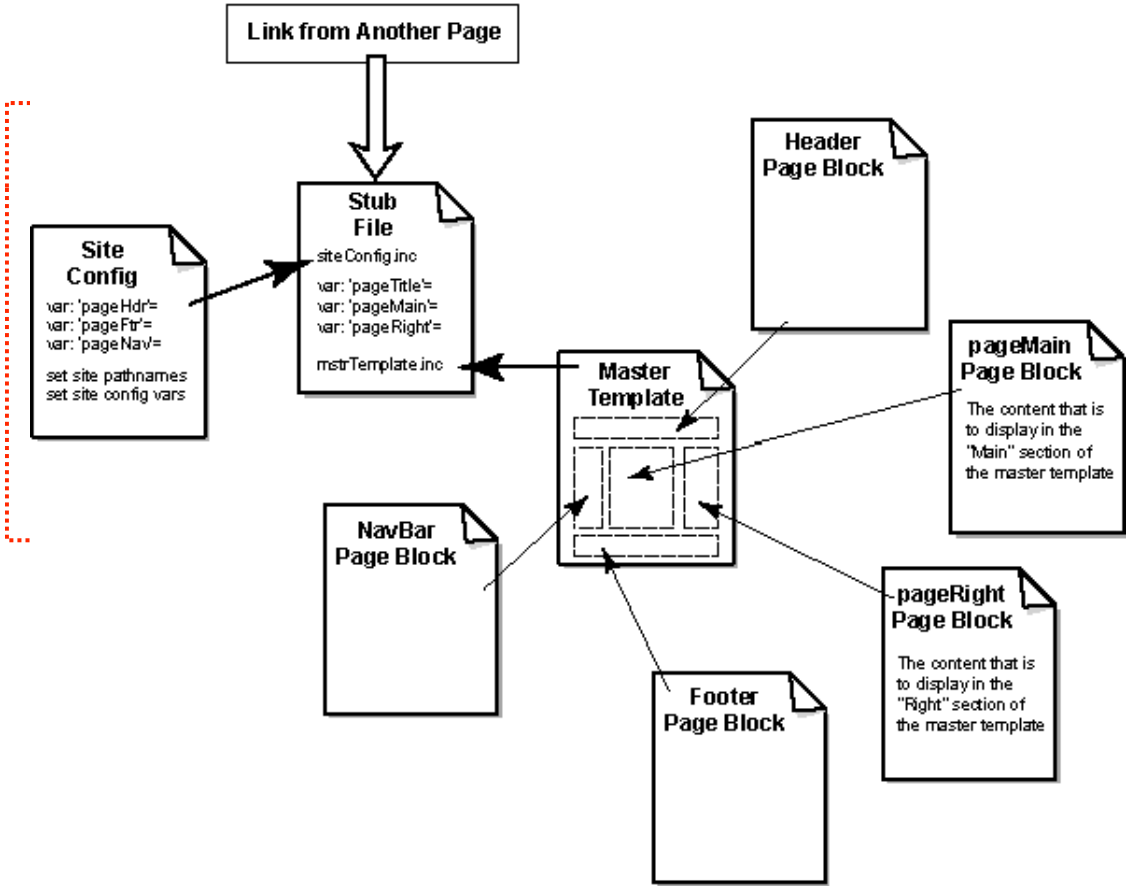
Template

```
<html>
<body>
<table>

[incl:$leftBlock]
[incl:$mainBlock]
```

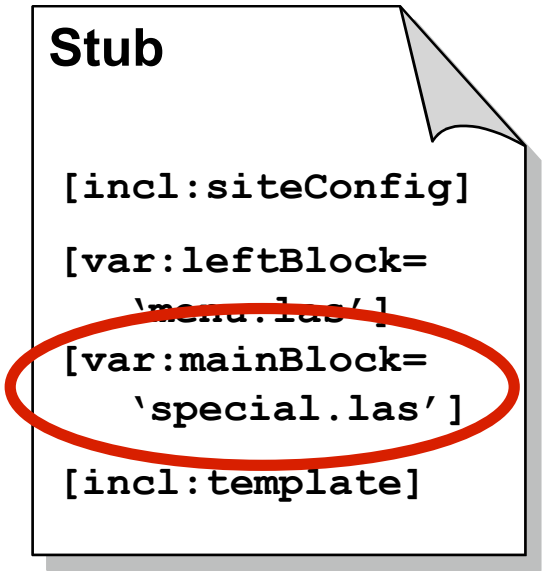
* pseudo code, proper LDML syntax not shown

LassoWARE

Corral File Relationships



Loading a Page: Classic Corral



- Must we manually define the block to be loaded?
- Q: how can the template figure out the content to display?
- A: standardized structural naming conventions.

those of you using databases to define stubs...
... keep quiet for a bit :-)

Structure: Beyond the Method

Depending on things being where they are supposed to be

Step 2: Structured File Names

section_page_block

/about

about_genl.lasso

about _genl_left.lasso

about _genl_main.lasso

/prod

prod_intro.lasso

prod_intro_main.lasso

stub file, so
no block name

The Structured Template

Header — defined in siteConfig by site/section as needed

about_genl_left.las

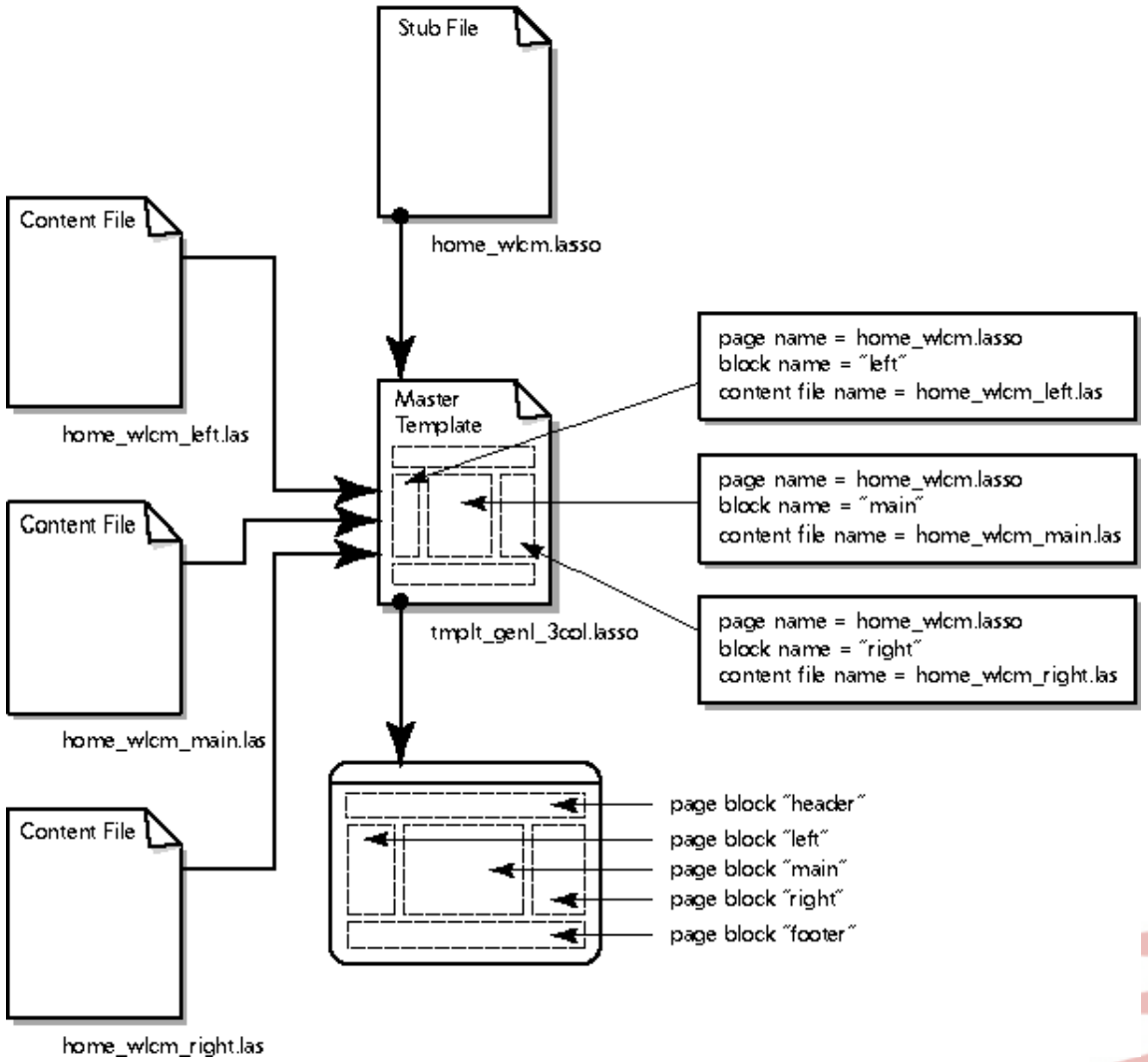
about_genl_main.las

about_genl_right.las

Footer — defined in siteConfig by site/section as needed

Step 2b: Structured File Names

- The template is now structured with each placeholder having a specific name
- Content files are named with respect to the placeholder they are to be used in



SiteConfig + Structured Names

- c_myName: name of current page
- c_myFldr: folder name for current page
- c_myPath: compete path to current page
- Where stub = /prod/widgets/widgets_intro.lasso
  - c_myName = widgets_intro.lasso
  - c_myPath = /prod/widgets/


```
var:'fw_pgBlock'=(string_replace:
   $c_myName,
   -find='.lasso',
   -replace='_main.lasso');

include:$c_myPath + $fw_pgBlock;
```

SiteConfig + Structured Names

- What if there is no widget_intro_left.lasso?

```
var:'fw_pgBlock'=(string_replace:
   $c_myName,
   -find='.lasso',
   -replace='_left.lasso');
if:(file_exists:$c_myPath + $fw_pgBlock);
   include:$c_myPath + $fw_pgBlock;
/if;
```

- This keeps the template general purposed for pages that make use of all or only some page blocks

The Template

```
if: (file_exists:$c_myPath + $fw_pgHdr);
   include:$c_myPath + $fw_pgHdr;
/if;
```

```
var:'fw_pgBlock'=
   (string_replace:
      $c_myName,
      -find='.lasso',
      -replace='_left.lasso');
if:
   (file_exists:
      $c_myPath +
      $fw_pgBlock);
   include:$c_myPath +
      $fw_pgBlock;
/if;
```

```
var:'fw_pgBlock'=
   (string_replace:
      $c_myName,
      -find='.lasso',
      -replace='_main.lasso');
if:
   (file_exists:
      $c_myPath +
      $fw_pgBlock);
   include:$c_myPath +
      $fw_pgBlock;
/if;
```

```
var:'fw_pgBlock'=
   (string_replace:
      $c_myName,
      -find='.lasso',
      -replace='_right.lasso');
if:
   (file_exists:
      $c_myPath +
      $fw_pgBlock);
   include:$c_myPath +
      $fw_pgBlock;
/if;
```

Assuming extra features / block, there is too much redundancy

(imagine several templates)

```
if: (file_exists:$c_myPath + $fw_pgFtr);
   include:$c_myPath + $fw_pgFtr;
/if;
```

Bring in the Suits

- Each page block for each template has identical code except the block name
- Make the block name a variable
- Put common code in an include named “blockManager”

```
var:'fw_pgBlock'=
  (string_replace:
    $c_myName,
    -find='.lasso',
    -replace='_left.lasso');
if:
  (file_exists:
    $c_myPath + $fw_pgBlock);
  include:$c_myPath + $fw_pgBlock;
/if;
```

Bring in the Suits

- Each page block for each template has identical code except the block name
- Make the block name a variable
- Put common code in an include named “blockManager”

```
var:'fw_blokName'='left';
include:'blockMngr.inc';
```

```
var:'fw_pgBlock'=
  (string_replace:
    $c_myName,
    -find='.lasso',
    -replace='_' +
        $fw_blokName +
        '.lasso');
if:
  (file_exists:
    $c_myPath + $fw_pgBlock);
  include:$c_myPath + $fw_pgBlock;
/if;
```

Improved Template

```
if: (file_exists:$c_myPath + $fw_pgHdr);
   include:$c_myPath + $fw_pgHdr;
/if;
```

Much better!

```
var:'fw_blokName'='left';
include:'blockMngr.inc';
```

```
var:'fw_blokName'='main';
include:'blockMngr.inc';
```

```
var:'fw_blokName'='right';
include:'blockMngr.inc';
```

blockMngr.inc
common stuff
to all blocks

```
if: (file_exists:$c_myPath + $fw_pgFtr);
   include:$c_myPath + $fw_pgFtr;
/if;
```

blockManager

- **blockManager contains all the peripheral code around a block**
- **this could be very little, or could include security, style options, content choice, etc.**

Example blockManager:

```
load user profile;
  set prefc vars;
  block require auth?;
    is user auth'd?
      include pageBlock;
    report auth errors;
  include pageBlock;
report profile errors;
```

blockLoader

- What if blocks have different needs?
  - 6 blocks, but only 3 do X
- Yet another layer...

Constant for your environment

```
load user profile;
  set prefc vars;
  block require auth?;
    is user auth'd?
      incl blockLoader;
    report auth errors;
  incl blockLoader;
report profile errors;
```

Potentially unique to each app

```
if:$fw_blokName == 'left';
  ...do lefty stuff...
  ...include:$fw_blokName
else:$fw_blokName == 'main';
  ...do main stuff...
  ...include:$fw_blokName
else:$fw_blokName == 'specials';
  ...sell armadillo...
else...;
/if;
```

LassoWARE

Improved Template

```
if: (file_exists:$c_myPath + $fw_pgHdr);
   include:$c_myPath + $fw_pgHdr;
/if;
```

```
var:'fw_blokName'='left';
include:'blockMngr.inc';
```

```
var:'fw_blokName'='main';
include:'blockMngr.inc';
```

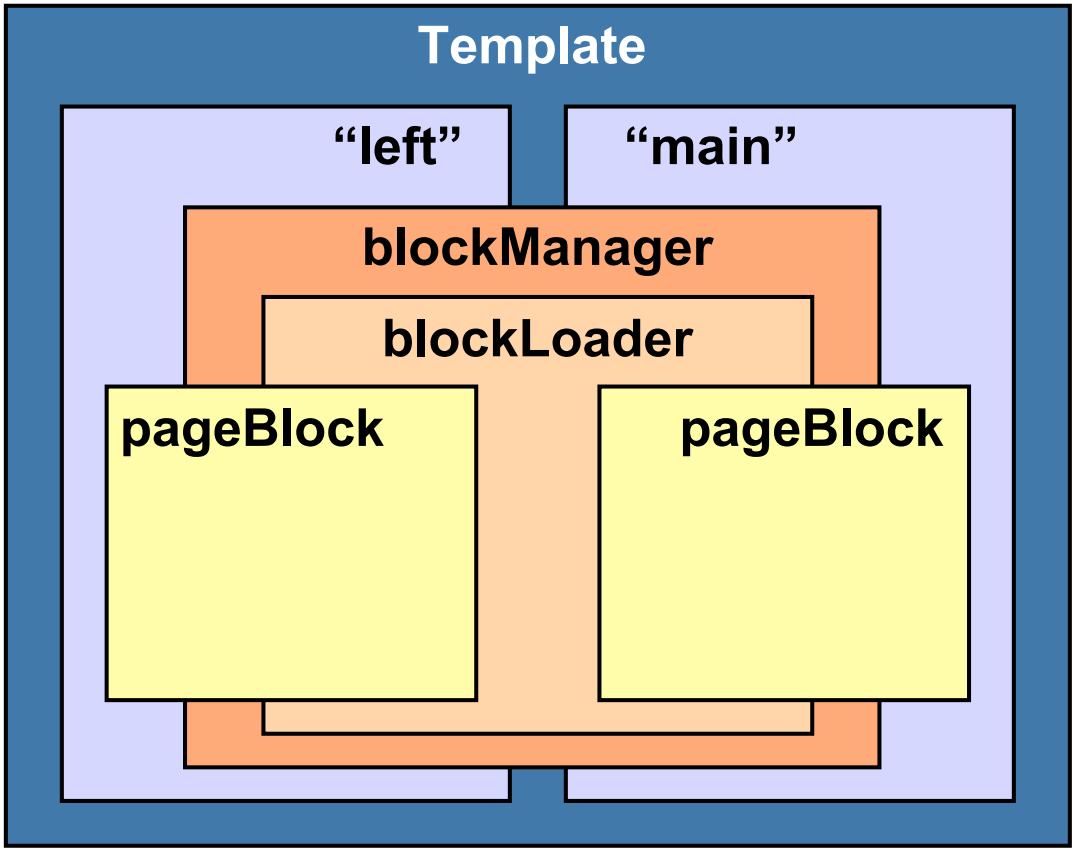
```
var:'fw_blokName'='right';
include:'blockMngr.inc';
```

blockMngr.inc
common stuff
to all blocks

blockLodr.inc
stuff unique to
specific blocks

```
if: (file_exists:$c_myPath + $fw_pgFtr);
   include:$c_myPath + $fw_pgFtr;
/if;
```

Page Loading Overview



Page Loading Overview

- Start with structured file naming
- Template defines block name, includes blockManager
- blockManager calculates content file name, performs general pre-content processing, includes blockLoader
- blockLoader performs block-specific pre-content processing, includes pageBlock content file
  - may or may not be integral to blockMngr

RAD Impact on Page Loading

- **Was:** define each page block manually
- **Is:** just name file a certain way, blocks are loaded automatically
- **How:** abstracted the template from loading a specific var refc to loading a generic object
- **Pros:** less tedium, eliminated redundancy
- **Cons:** two extra includes; negligible

More Importantly

- Critical analysis
  - we looked at a repetitive task and broke it into unique / redundant pieces
- Created a RAD toolset
  - gave the code “awareness”
  - made the code do something we did
  - general purpose
  - reusable, non-redundant
- Going to use this process over & over

Managers: Divide and Conquer

In RAD, middle managers are a good thing. Honest.

Managers vs Includes

- What makes a manager?
- www.dictionary.willits:
  - makes environment driven decisions
  - makes configuration driven decisions
  - abstracts / distills complex operations
  - integrates multiple functions / routines
  - no assumptions about location, content...
  - independent of other application functions

Manager Tasks

- Page block loading
- Database actions
- User authentication
- Error management
- Input validation
- Content integration

Database Actions

- More than just -add, -update...
- Integregated flow of
  - record locking
  - input validation
  - database action
  - error handling
  - action confirmation / user feedback

Database “actionShells”

- Integrated nested routines
- One for each unique action
  - view
  - add
  - addOK
  - update
  - updateOK
  - delete
  - deleteOK
  - select

Each action shell has a unique combination of peripheral functions

actionShell : Update Prep

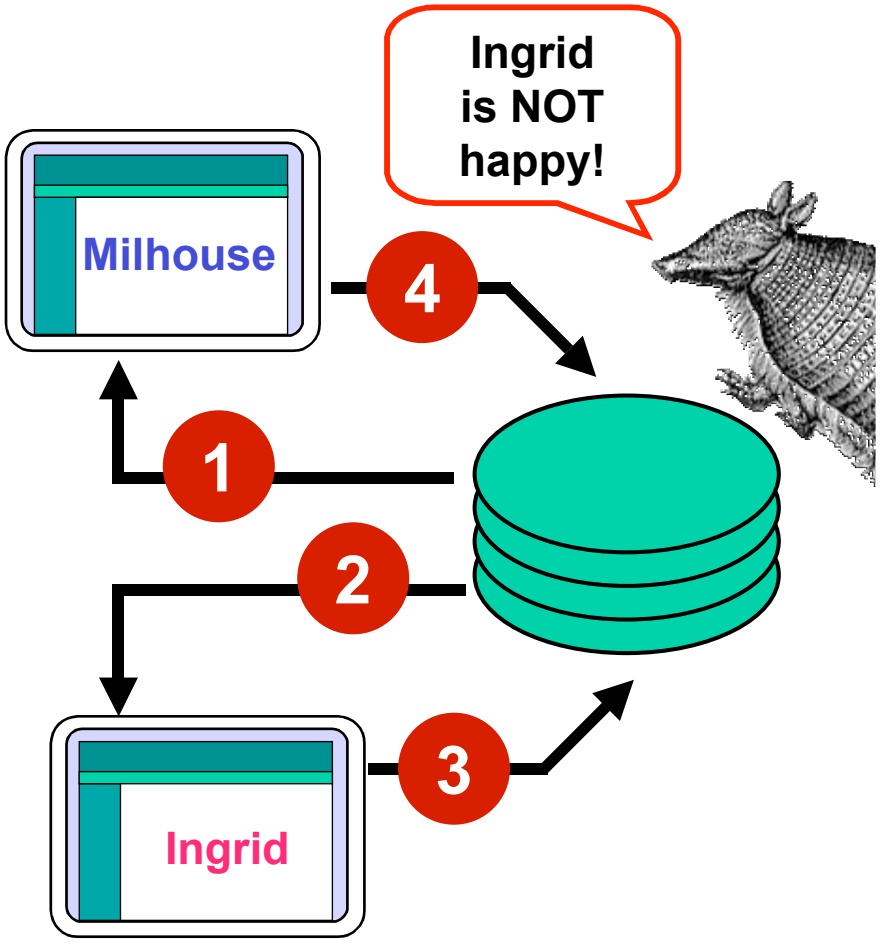
- Verify a recordID value exists
  - Search for the current record
  - Test for errors caused by the search
    - Is record currently locked? No. Continue
      - Generate a lockID
      - Write lockID to record to prevent another user from making changes to the record while this user prepares edits
      - Test for errors caused by writing the lockID
        - Search for record and retrieve needed fields
        - Test for errors
          - Include the update form pageBlock
        - Report Errors
      - Report Errors
    - Report record is locked
  - Report Errors
- Report No Record Number Error

actionShell : UpdateOK

- Convert action_params to vars (done in siteConfig)
- Load input validation rules for current database
- Test for errors reported by validation include
    - Verify a recordID exists
        - Validate that the record lock is valid
            - Read table definition file
            - Read array of form input params
            - Create corresponding array of field names
            - Build SQL UPDATE statement from arrays
            - Execute SQL statement
            - Test for update action errors
                - Search for record to acquire field data for display
                - Test for errors
                    - Construct response page HTML
                - Report Errors
            - Report Errors
        - Report lock expiration or invalid session
    - Report no record ID error
- Display Input Error Messages

Record Locking

- Milhouse loads record #991 to update
- Ingrid loads record #991 to update
- Milhouse has to chase an armadillo out of the back yard.
- Ingrid, updates the record; heads out on a date
- Milhouse returns, updates the record
  - wipes out Ingrid's changes
- Ingrid returns from a bad date, check the db
  - changes are gone!
  - gives up on computers and men

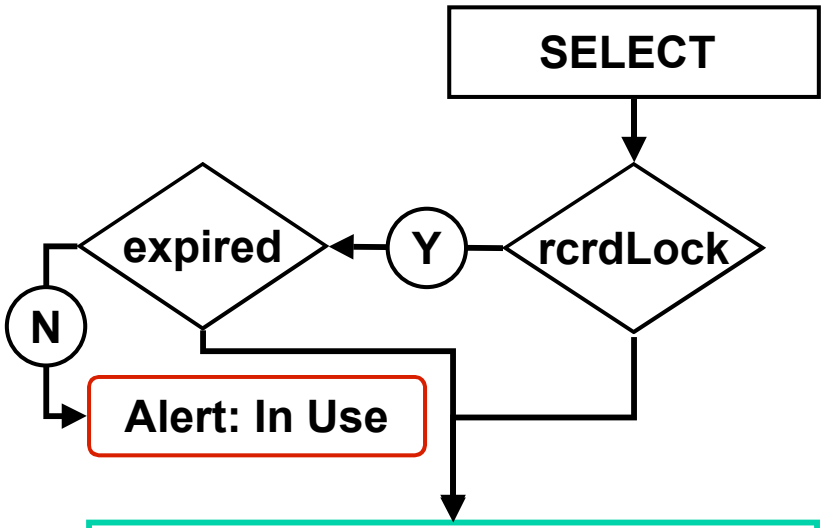


Record Locking

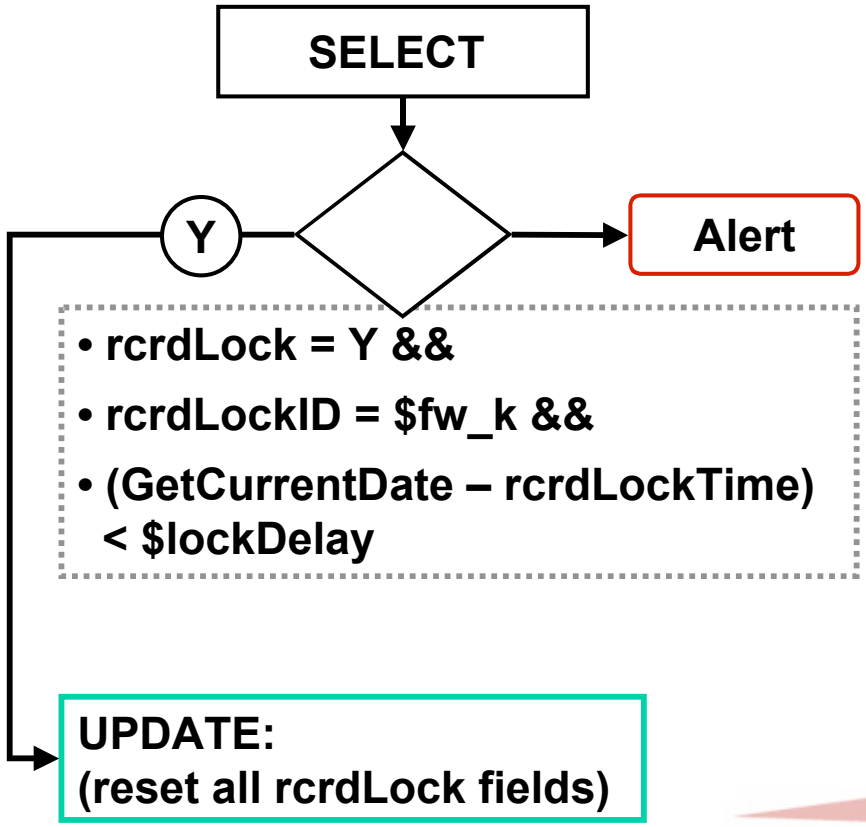
- Record locking fields:
  - rcrdLock = Y | empty
  - rcrdLockID = random session ID
  - rcrdLockTime = Lasso date & time
  - rcrdLockOwnr = user name (optional)
- $fw_gLockDelay = max lock duration (mins)
- Lock the record when loaded into an update form
- Unlock the record when update is submitted

Record Locking Process

Read Record



Update Record



LassoWARE

Locking a Record

```
[var:'fw_k'=(lasso_uniqueID)]

[inline:
     -username=$fw_gClientNm,
     -password=$fw_gClientPw,
     -database=$fw_dbName,
     -sql=(
          'UPDATE ' + $fw_dbName + '.' + $fw_dbTable + ' SET ' +
          'rcrdLock="Y", ' +
          'rcrdLockTime="' + (Date_GetCurrentDate) + '", ' +
          'rcrdLockOwnr="' + $usrName + '", ' +
          'rcrdLockID="' + $fw_k + '" ' +
          'WHERE ' + $fw_dbPKey + '="' + $fw_dbKeyVal + '")]
```

Verify Lock Before Update

```
[inline:
   -username=$fw_gClientNm,
   -password=$fw_gClientPw,
   -database=$fw_dbName,
   -sql=('SELECT rcrdNo, rcrdLock, rcrdLockID,
      rcrdLockTime, rcrdLockOwnr ' +
      ' FROM ' + $fw_dbName + '.' + $fw_dbTable +
      ' WHERE ' + $fw_dbPKey + '="' + $fw_dbKeyVal + '"')]

      [if: (error_currentError)==(error_noError) && (found_count) != 0]

         [var:'fw_lockDelta'=(Date_Difference:
            (Date_GetCurrentDate),
            (field:'rcrdLockTime'),
             MinutesBetween)]

         [if:((field:'rcrdLock') == 'Y') &&
            ((field:'rcrdLockID') == $'fw_k') &&
            ((var:'fw_lockDelta') < $'fw_gLockDelay')]

         # all is good, we can proceed with the Update
```

Automating SQL Statements

- INSERT and UPDATE
- Converting inputs to vars
- Mapping variables to fields
- Determining which fields involved
- Constructing the SQL statement

Typically all hand coded

autoSQL : Convert Inputs

- Convert action_params to vars
  - done from siteConfig so its automatic

```
if: (Client_FormMethod)=='POST';
  loop: (action_params)->size;
    var:((action_params)->(get:loop_count)->first)=
    ((action_params)->(get:loop_count)->second);
  /loop;
/if;
```

autoSQL : Field Map

- Map database fields to input fields
- Whole database in a single table
- Every form in application uses same input name for a given field / database

```
[output_none]
  apv = rcrdAppvd
  ownr = rcrdOwnr
  grp = rcrdGroup
  rdy = artclReady
  dat = artclDate
  ttl = artclTitle
  athr = artclAuthor
  cat = artclCategory
  lyt = artclLayout
  syn = artclSynopsis
  cpy = artclStory
[/output_none]
```

autoSQL : Field Map

- Load field map into arrays
  - $fw_paramNames, $fw_fieldNames

```
// load the file
$fw_tableDefn=(file_read:($fw_filePath))->(split:'\r');

// split into arrays
loop: ($fw_tableDefn->size);
   var:'fw_fieldDefn'=($fw_tableDefn->(get:loopcount));
   if:($fw_fieldDefn == '')||($fw_fieldDefn >>'output_none');
   else;
      $fw_paramNames->
         (insert:($fw_fieldDefn->(split:'=')->(get:1)));
      $fw_fieldNames->
         (insert:($fw_fieldDefn->(split:'=')->(get:2)));
   /if]
/loop]
```

autoSQL : so far...

- Now we have:
  - variables = input field names
  - an ordered array of input field names
  - an ordered array of database field names
- We can now easily correlate
  - value ↔ name ↔ field
  - ‘Army Dillo’ ↔ $athr ↔ field:artclAuthor

autoSQL : Build UPDATE (i)

```
// the basics
var:'fw_sqlActn'=(string);
$fw_sqlActn = 'UPDATE ' + $fw_dbName + '.' + $fw_dbTable;
$fw_sqlActn += (' SET ');
$fw_sqlActn += ('rcrdModified="' +
   ((date_format:
      (Date_GetCurrentDate),
      DateFormat='%Y-%m-%d %T') +
   ' - ' + (var:'usrName')) + '", ');

// reset the record lock fields
$fw_sqlActn += ('rcrdLock="' + '", ');
$fw_sqlActn += ('rcrdLockTime="' + '", ');
$fw_sqlActn += ('rcrdLockOwnr="' + '", ');
$fw_sqlActn += ('rcrdLockID="' + '", ');
```

A standard thing
I do in all tables to
stamp the last update.
Multi-line history log
version also possible

autoSQl : Build UPDATE (ii)

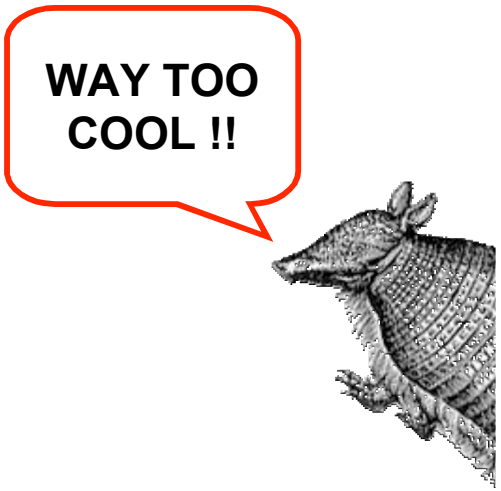
```
// loop through the table definition arrays and see if
// a particular field was submitted by the form
// we do that by reading the array and seeing if a var with
// name exists, if so, include it in the SQL statement

loop: ($fw_paramNames->size);
   if: (var_defined:$fw_paramNames->(get:loop_count));
      $fw_sqlActn +=
         (($fw_fieldNames->get:loop_count) + '="' +
         (var:($fw_paramNames->get:loop_count))** + '", ');
   /if;
/loop;

// we end up with a trailing comma that has to be deleted
$fw_sqlActn->(removetrailing:', ');

// finally add the reference to the keyfield
$fw_sqlActn += (' WHERE '+$fw_dbPKey+'="'+$fw_dbKeyVal+'"');
```

LassoWARE

** use this: (string_replace: (var:($fw_paramNames->get:loop_count)), -find='\"', -replace='\"\"')

autoSQl : Build UPDATE (iii)

```
// all that's left to do is execute the statement:
inline:
     -username=$fw_gClientNm,
     -password=$fw_gClientPw,
     -database=$fw_dbName,
     -sql=$fw_sqlActn;
```



- And that's how to automate SQL
- Want to add a field to a form?
  - just add it — nothing else to do!
- Want to add field to a table?
  - just add input=fieldName in text file

actionShell : Other Functions

- Error management: refer to the paper
- Input validation: refer to paper
- Automate action acknowledgement
  - response page to actionShell has no content file
  - standardized include builds general purpose string and button to link to a “continue” page
  - override possible by creating a content file

actionShell Manager

- Multiple shells, peripheral includes
- Use “actionVars”
- actionManager is built into my blockLoader
  - reads $fw_actnNm and auto includes the actionShell
- actionVars are all I need to program db actions

```
[var:
    'fw_actnNm'='(1)',
    'fw_actnInline'='',
    'fw_actnBlock'='',
    'fw_actnContinue'='',
    'fw_actnDb'=$fw_***Db,
    'fw_actnTbl'=$fw_***Tbl,
    'fw_actnFlds'='',
    'fw_actnPKey'='',
    'fw_actnKeyVal'='',
    'fw_actnWhere'='',
    'fw_usrPrvlg'='(2)',
    'fw_actnCnfrmFld'='',
    'fw_actnCnfrmStyle'='body']

(1) add,addOK,update,updateOK,
    delete,deleteOK,view
(2) usrVw,usrAdd,usrUpdt,usrAppv,
    usrDlt,usrLvl1,usrLvl2,
    usrLvl3,usrLvl4,usrExec
```

autoSQL : Related Records

- Well, I'm still working on that one
- Probably something like this:
  - convert db & tbl vars to arrays
  - easy to build table.field formats using the field maps
  - alter the assembly process accordingly

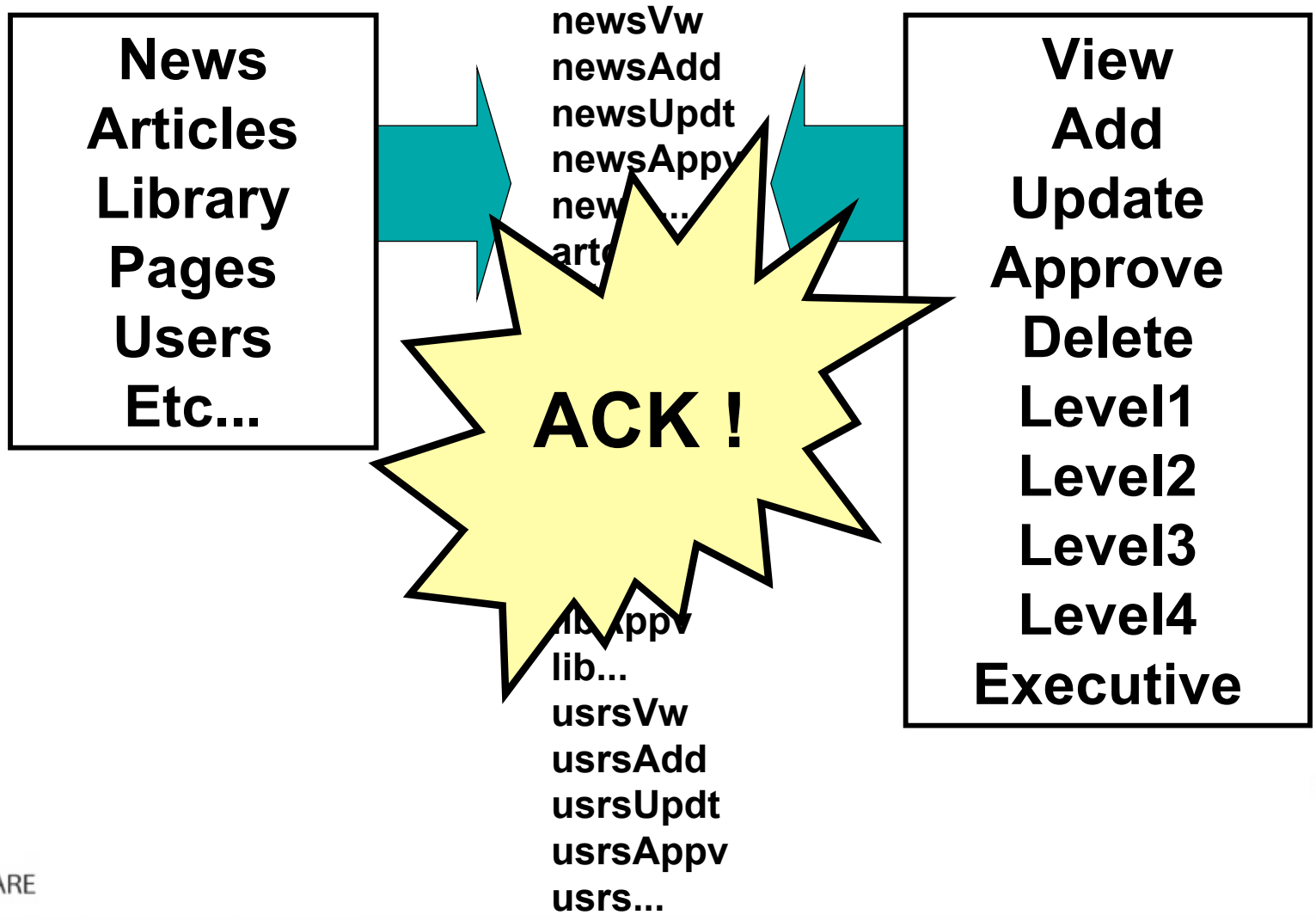
User Authentication

- User profile
- User permissions
  - multiple secured work spaces
  - multiple permissions / space
- Dynamic apps = dynamic spaces / permissions
- Need UA manager that adapts automatically

Pre-History (Last Week)

- Users table with a field for every permission
- User getProfile routine with dynamic work space name, but hard privilege names
- Admin pages with hard coded matrix table
- Adding a work space or privilege involved several file updates
  - a tedious chore for new projects

What I Usually Need



Automating Permission Maps

```
// modules (work spaces) : codeName=friendlyName
[var:'fw_authModsList'=(array:
  'usrs'='Users',
  'artcls'='Articles',
  'code'='Code',
  'news'='News')]

// permissions per module
[var:'fw_authPermsList'=(array:
  'usrVw'='View',
  'usrAdd'='Add',
  'usrUpdt'='Update',
  'usrApv'='Approve',
  'usrDlt'='Delete',
  'usrLvl1'='Level1',
  'usrLvl2'='Level2',
  'usrLvl3'='Level3',
  'usrLvl4'='Level4',
  'usrExec'='Executive')]
```

Storing Permissions Matrix

- Want only one table field to accommodate new features / permissions
- Still need to retrieve all or specific modules, all or specific permissions
- Data structure:
  - map of multiple (map of (pairs))

```
map:
  (news)=(map: (usrDlt)=(N), (usrAdd)=(N),...)),
  (artcls)=(map: (usrDlt)=(N), (usrAdd)=(N),...)),
  (usrs)=(map: (usrDlt)=(N), (usrAdd)=(N),...))
```

Creating Form Tables

- Matrix of mods/perms
- Loop and write HTML table code
- Inside each cell, insert a popup/radio btns
- Form input name = permName_modName
- Add form; default=N
- Update default = value

Table Formatting Vars:

```
[var:
   'fw_permsMode'='',
   'fw_permsTblBorder'='0',
   'fw_permsTblPadding'='2',
   'fw_permsTblSpacing'='1',
   'fw_permsLblColor'='009999',
   'fw_permsLblStyle'='bodyboldwhite',
   'fw_permsCellColor'='#FFDEAD',
   'fw_permsCellColor2'='#FFCC99',
   'fw_permsCellStyle'='body',
   'fw_permsCellWidth'='120',
   'fw_permsCellHeight'='']
```

usrAdd_code

	Users	Articles	Code	News
View	Yes No	Yes No	Yes No	Yes No
Add	Yes No	Yes No	Yes No	Yes No
Update	Yes No	Yes No	Yes No	Yes No
Delete	Yes No	Yes No	Yes No	Yes No
Approve	Yes No	Yes No	Yes No	Yes No
Level-1	Yes No	Yes No	Yes No	Yes No
Level-2	Yes No	Yes No	Yes No	Yes No
Level-3	Yes No	Yes No	Yes No	Yes No
Level-4	Yes No	Yes No	Yes No	Yes No
Executive	Yes No	Yes No	Yes No	Yes No

LassoWARE

Consolidating the Table Inputs

```
var:'fw_authUsrPerms'=(map);
loop: $fw_authModsList->size;
  local:'fw_modName'=($fw_authModsList->get:loop_count);
  local:'fw_usrPerms'=(map);

// create the inner maps of user permissions
// values come from form fields which have been
// converted to vars by glbl_getForm which are named
// permName_modName such as userVw_news
  loop: $fw_authPermsList->size;
    local: 'fw_permName'=($fw_authPermsList->get:loop_count);
    #fw_usrPerms->(insert: #fw_permName =
      (var:#fw_permName + '_' + #fw_modName));
  /loop;

// create outer map of modules, insert the inner maps
  $fw_authUsrPerms->(insert: #fw_modName = #fw_usrPerms);
/loop;
```

This was from a version prior to friendly names. This would have to be updated to pull the first element of the pairs as shown a few slides ago

autoPermissions

- Adding new modules / permissions
  - just add to the two master arrays
- Updating a user automatically adds new perms
  - table is draw from the arrays
  - if no value found, default to N
- Users not updated remain compatible
- No maintenance of admin forms, getProfile code, database table

What Did He Say?

- Software managers for:
- Page block loading
- Database actions
- User authentication
- Error management
- Input validation
- Content integration

Modules: Plug and Play Corral

Lasso community still lacks a truly interchangable code specification

Modular Corral / FrameWork

- Plug & play reusability of application code
- Auto configure to environment styles
- Global and local resources
  - non-standard innards, standard hooks
- Communication between modules to integrate information and skills (“services”)
- Self contained configuration, content, data management, administration, resources
- Constructed of Objects and Types

Module Structure

Site Structure

/artcls
/code
/news
/site
    /controls
    /css
    /files
    /imgs
    /libs
    /msthd
    /srvcs
    /tags
    siteConfig.inc
/siteMngr
/siteUsrs

Module Structure

/artcls
    _modConfig.inc
    /_admn
    /_resources
        /css
        /imgs
        /msthd
            /msthdImgs
            /templates
        /libs
        /srvcs
            getMyConfig.inc
    artcls_intro.lasso
    artcls_intro_main.las
    artcls_intro_left.las

siteConfig and Modules

```
var:
   'c_gMngrPath'='/siteMngr/',

   'c_gLibsPath'='/site/libs/',
   'c_gSrvcsPath'='/site/srvcs/',
   'c_gImgsPath'='/site/imgs/',
   'c_gFilesPath'='/site/files/',

   'c_gMsthdPath'='/site/msthd/',
   'c_gMsthdImgsPath'='/site/msthd/imgsMsthd/',
   'c_gTmpltsPath'='/site/msthd/templates/',
```

Global Resources

'c_gLibsPath'

```
var:
   'c_mAdmnPath'=$c_modPath + '_admn/',

   'c_mLibsPath'=$c_modPath + '_resources/libs/',
   'c_mSrvcsPath'=$c_modPath + '_resources/srvcs/',
   'c_mImgsPath'=$c_modPath + '_resources/imgs/',
   'c_mFilesPath'=$c_modPath + '_resources/files/',

   'c_mMsthdPath'=$c_modPath + '_resources/msthd/',
   'c_mMsthdImgsPath'=$c_modPath + '_resources/msthd/imgsMsthd/',
   'c_mTmpltsPath'=$c_modPath + '_resources/msthd/templates/',
```

Local Resources

'c_mLibsPath'

siteConfig and Modules

```
// acquire the section config information

inline:
   -username=$fw_gClientNm,
   -password=$fw_gClientPw;

   if: (file_exists: ($c_modPath + '_modConfig.lasso'));
      library: ($c_modPath + '_modConfig.lasso');
   else;
      ..... error managment .....
   /if;
/inline;
```

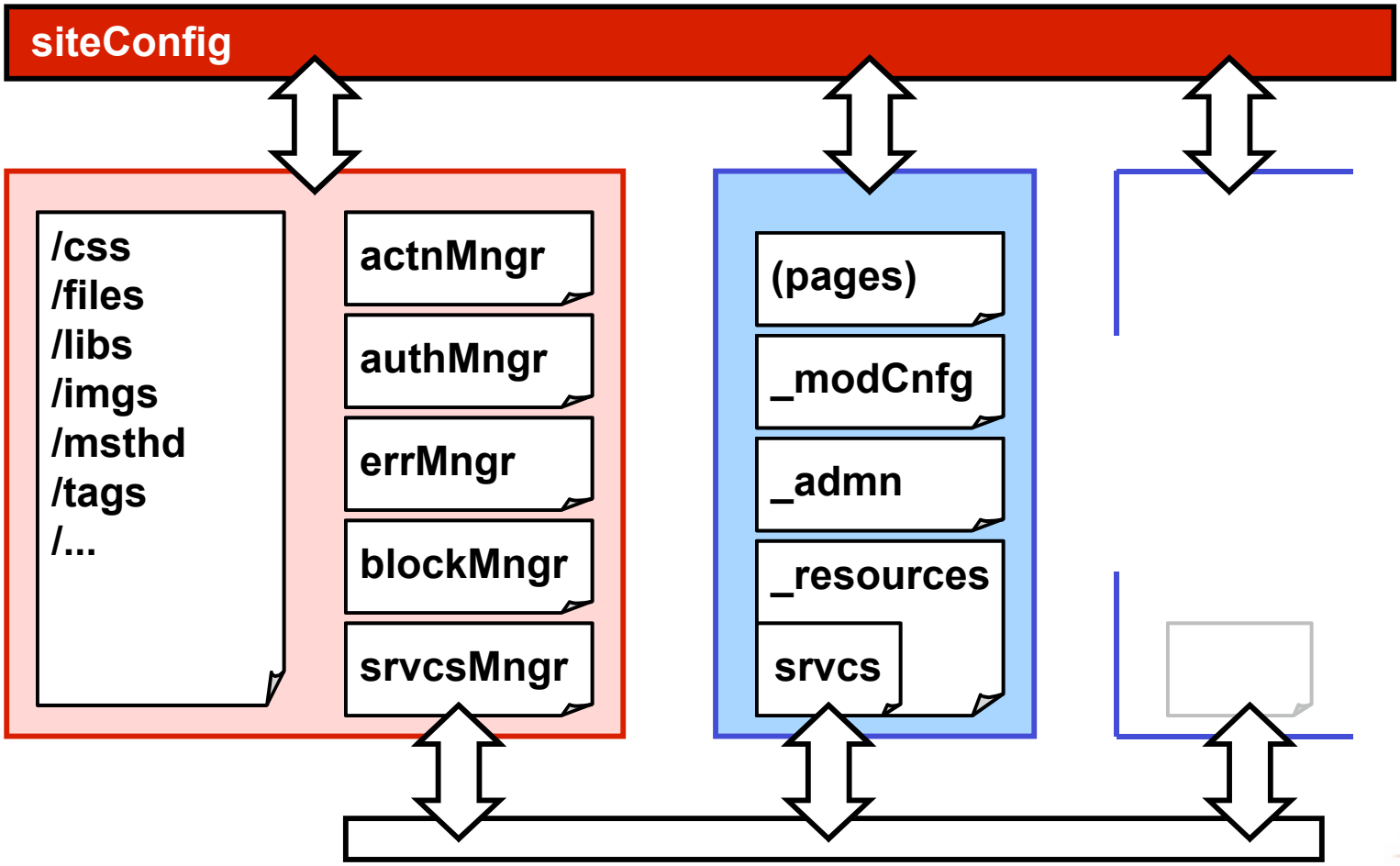
Typical modConfig

```
// modCode is used to build a number of internal references
var:'fw_modCode'='artcls';
// pageAuth triggers whether the page needs authorization
   if: $c_myPath == $c_modPath;
      var:'fw_pageAuth'='N';
   else: $c_myPath >> '_admn';
      var:
         'fw_pageAuth'='Y',
         'fw_mnuDefns'='',
         'fw_pgTmplt'='a1col',
         'c_hdrFile'='hdr_siteMngr.las',
         'c_ftrFile'='ftr_siteMngr.las';
   /if;
// get module user configuration info from a config dbTable
var:'fw_srvcParams'=(map:
   'srvcFldr'=$fw_modCode,
   'srvcName'='getMyConfig.inc');
include: ($c_gLibsPath) + 'fw_glbl_srvcMngr.inc';
```

Module Services

- Service — a context independent include
- Service manager
  - abstracts calling mechanism from location and environment specifics
  - verifies service available
  - cleans up vars to avoid crosstalk

Basic Modular Scheme

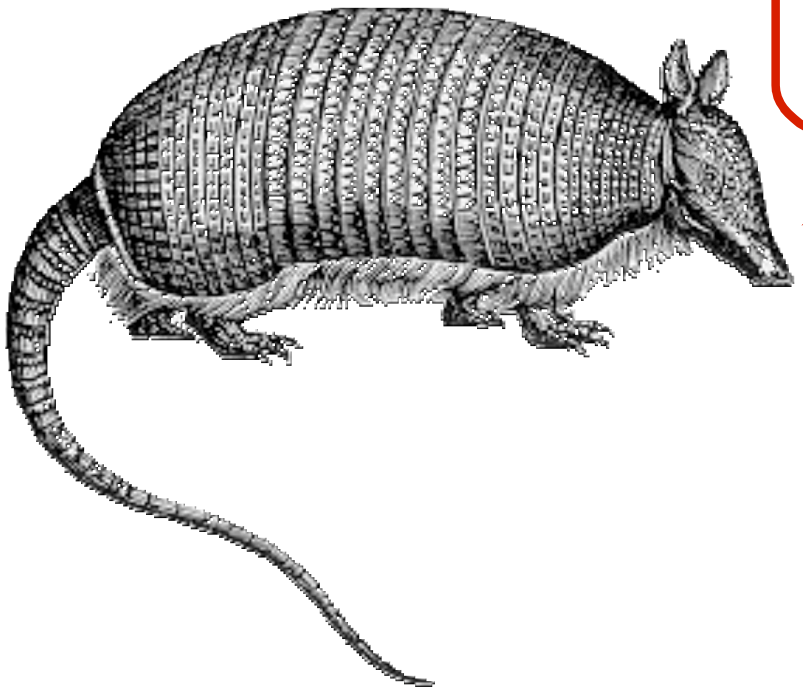


[FrameWork: -Pro]

- Standardized
  - architecture, libs, name spaces, process managers
  - high level interfaces to automated systems
- Modular
  - separation of content & code
  - global and local resource access
  - extensible and interchangable
- Adaptable to multiple Corral types (at module level)
- Soon to be a development “environment” with utilities, glossaries, starter kit modules

Why were you here?

- To learn more about Armadillos
- Understand RAD philosophy, systems
- Learn new ways to look at your code
  - break it up, standardize it, automate it
- I hope I gave you some ideas



Thanks for Listening