ETRACS

Intermediate Systems Training

Basic Refresher Course

July 17 – 21, 2023

Rameses Systems Inc

One Central Hotel
Cebu City, Cebu

Basic Refresher Course

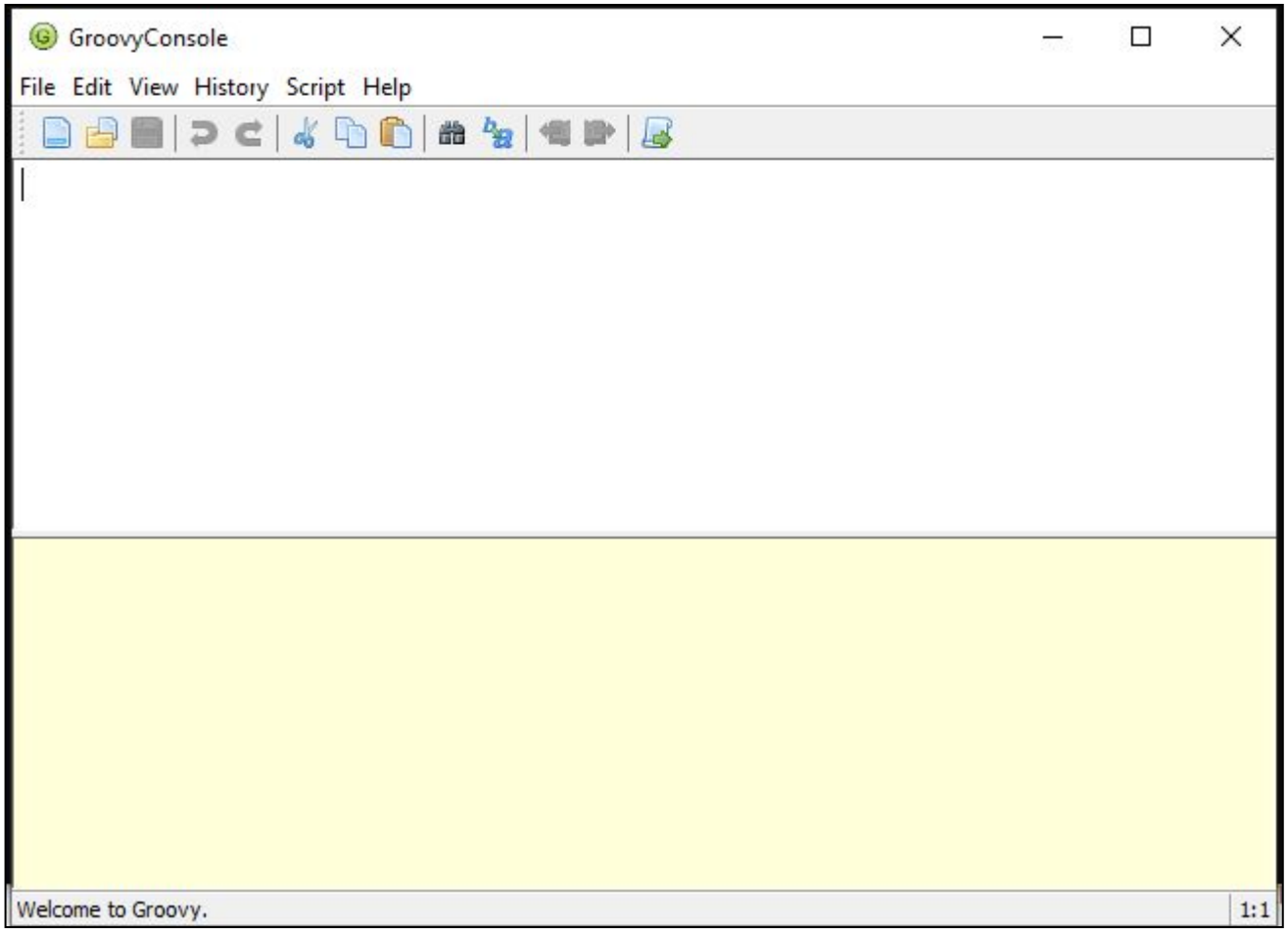
- Groovy Language
- XML Language
- Osiris3 Platform

GROOVY LANGUAGE REFRESHER

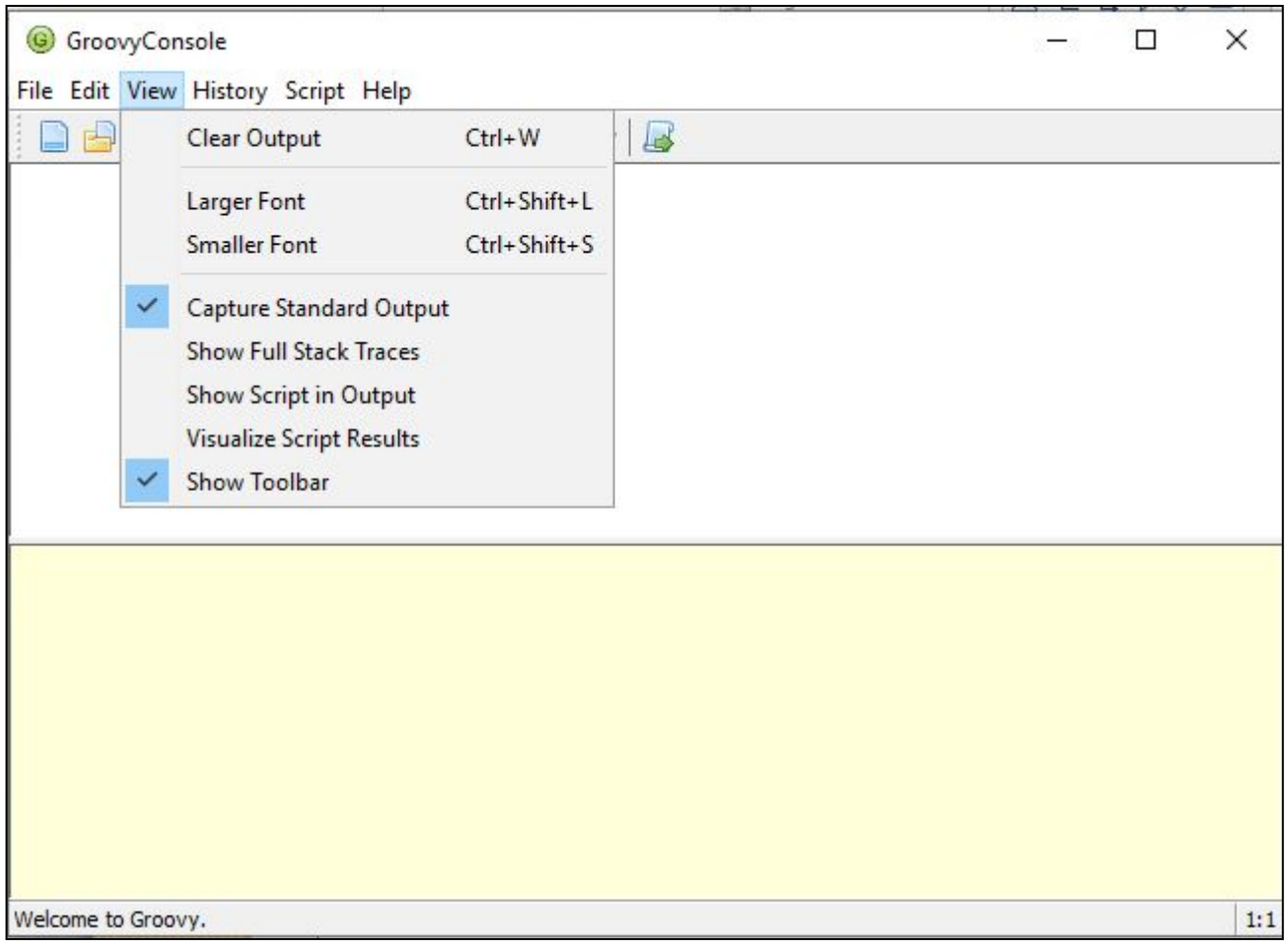
The Groovy Console

- Simple IDE to execute Groovy codes
- Use to test code snippets

Launching Groovy Console



Groovy Console Settings



Groovy Console Commands

- **Ctrl R** - execute script
- **Ctrl W** - clear output window
- **Ctrl SHIFT L** - larger font
- **Ctrl SHIFT S** - smaller font

Operators

```
-----------------------------------
Arithmetic Operators:
-----------------------------------
Addition:                  +
Subtraction:               -
Multiplication:            *
Division:                  /
Modulus:                   %
Increment:                 ++
Decrement:                 --

-----------------------------------
Assignment Operators:
-----------------------------------
Simple Assignment:
  =

Compound Assignment:
  +=          x = x + n
  -=          x = x - n
  *=          x = x * n
  /=          x = x / n
  %=          x = x % n
```

```
-----------------------------------
Comparison Operators:
-----------------------------------
Equal to:                  ==
Not equal to:              !=
Greater than:              >
Less than:                 <
Greater than or equal to:  >=
Less than or equal to:     <=

-----------------------------------
Logical Operators:
-----------------------------------
Logical AND:    &&
Logical OR:     ||
Logical NOT:    !

-----------------------------------
Conditional Operators:
-----------------------------------
* If Statement
  if (expr) { statements }

* If-Else Statement
  if (expr) {
    statements
  } else {
    statemetns
  }

* Ternary Operator:
  condition ? true-expr : false-expr
```

Data Types

- Supports all Java data types
- No primitive data types
- Dynamic and Strong Typing
- Lists and Maps are treated as basic types

Variable Declaration Example

```
//strong typing
int qtyIssued = 100
String name = "Juan"

//dynamic type
def amount = 1500.00
def company = "Rameses"
```

String

- Enclosed by either:
  - (‘…’) – single line
  - (‘’’…’’’) – multi-line

```
def name = 'Juan dela Cruz'

def html = '''
    <html>
        <body>
        </boxy>
    </htm>
'''
```

Closures

Closure

- a block of code assigned to a variable
- can access variables within its scope

```
String name = 'Juan';

def sayHello = { println 'Hello ' + name}
sayHello();

name = 'Peter';
sayHello();
```

```
Hello Juan

Hello Juan
Hello Peter
```

Closure with Parameters

- parameters are listed before the **->** token

```
def printSum = { a, b ->
    print a + b
}

printSum( 7, 9)
```

```
16
```

Implicit Parameter

- **it** – if you have a Closure that takes a single argument, you may omit the parameter definition of the Closure

```
def sayHello = { println "Hello, ${it}..."  }
sayHello("World")
sayHello("Boxy")
```

```
Hello, World...
Hello, Boxy...
```

Iterations

- All Java Loop Structures are valid except Do-Wile

```
def grades = [79, 90, 100, 67, 89]

println 'traditional for loop -----------'
for( int i=0; i<grades.size(); i++ ) {
    print grades[i] + ' '
}

println '\nUsing generic ------------------'
for( int i : grades ) { print i + ' ' }

println '\nGroovy syntax-------------------'
for( i in grades) { print i + ' ' }
```

```
traditional for loop -----------
79 90 100 67 89
Using generic ------------------
79 90 100 67 89
Groovy syntax-------------------
79 90 100 67 89
```

Looping using Range

- uses the ".." syntax

```
println 'for-loop and range...'
for( i in 1..5) {
    print i + ' '
}

println '\nloop using range only'
(1..5).each {
    print '*'
}
```

```
for-loop and range...
1 2 3 4 5
loop using range only
*****
```

Loop using times() method

```
int row = 10
10.times {
    row.times { print '*'}
    row = row - 1
    println ""
}
```

```
**********
*********
********
*******
******
*****
****
***
**
*
```

List

- Java List is valid
- Groovy List is more concise

```
//standard java
List names = new ArrayList()
names.add("Juan")
names.add("Peter")

//groovy way
def countries = []
countries.add("PHIL")
countries.add("USA")

//declare and initialize
def units = ['in', 'cm', 'm', 'km']
```

Accessing List Element

- use normal **get()** method or the **[index]** operator

```
def grades = [90, 87, 95, 72, 56]

println grades

// using the [] operator
println 'Element 2 -> ' + grades[1]

// using the get() method
println 'Element 3 -> ' + grades.get(2)
```

```
[90, 87, 95, 72, 56]
Element 2 -> 87
Element 3 -> 95
```

Iterating the List Element

- Use the **each()** method

```
def grades = [90, 87, 95, 72, 56]

grades.each {
    println it
}
```

```
90
87
95
72
56
Result: [90, 87, 95, 72, 56]
```

Sorting List using **sort()**

```groovy
def grades = [90, 87, 95, 72, 56]

// using standard sort
grades.sort()
println grades

println "----------------------"
// specify sorting comparison
grades.sort{ a, b -> a - b  }
println 'asc  -> ' +  grades

grades.sort{ a, b -> (a-b) * -1 }
println 'desc -> ' + grades
```

```
[56, 72, 87, 90, 95]
----------------------
asc  -> [56, 72, 87, 90, 95]
desc -> [95, 90, 87, 72, 56]
```

Finding Elements: **find** and **findAll**

```
def grades = [90, 87, 95, 72, 56, 100]

// first item that matches the condition
int firstPassingGrade = grades.find{ it >= 75 }
println '1st Passing Grade -> ' + firstPassingGrade

// all items that match the condition
def passingGrades = grades.findAll{ it >= 75 }
println 'Passing Grades -> ' + passingGrades
```

```
1st Passing Grade -> 90
Passing Grades -> [90, 87, 95, 100]
```

Joining Elements using **join()**

```
def grades = [90, 87, 95, 72, 56, 100]

// display grades separated by comma
println grades.join(', ')
```

```
90, 87, 95, 72, 56, 100
```

Collecting Elements using **collect()**

```
def countries = ["Philippines", "Singapore", "Japan" ]
def initials = countries.collect{ it[0] }
println 'initials -> ' + initials

def countryLengths = countries.collect{
    "${it} (${it.size()})"
}
println 'Country Name Lengths '
countryLengths.each{ println  '   ' + it}
println ''
```

```
initials -> [P, S, J]
Country Name Lengths
  Philippines (11)
  Singapore (9)
  Japan (5)
```

Maps

- associative set of key and value pairs
- simple and elegant with the use of closures
- simple to create

Creating and Populating a Map

```
// java way
Map os = new HashMap()
os.put('Windows', 'Microsoft')
os.put('Linus', 'Linux')

// groovy way
def osg = [:]                    //declare an empty map
osg['Windows'] = 'Microsoft'     // using [] operator
osg.linus = 'Linux'              // using dot(.) operator

//create and initialize a map
def personnel = [
    idno     : '001',
    firstname: 'juan',
    lastname : 'dela cruz'
]
```

Accessing a Map Element

```
def personnel = [
    idno     : '001',
    firstname: 'juan',
    lastname : 'dela cruz'
]

// java get() method
println personnel.get('idno') + ' ' + personnel.get('firstname')

// the [] operator
println personnel['idno'] + ' ' + personnel['firstname']

// the dot(.) operator
println personnel.idno + ' ' + personnel.firstname
```

```
001 juan
001 juan
001 juan
```

Iterating Elements using **each()**

```
def personnel = [
    idno     : '001',
    firstname: 'juan',
    lastname : 'dela cruz'
]

personnel.each { key, value ->
    println "$key =  $value "
}
println ''
```

```
idno =  001
firstname =  juan
lastname =  dela cruz
```

OSIRIS3 PLATFORM REFRESHER

Osiris3 Platform Refresher

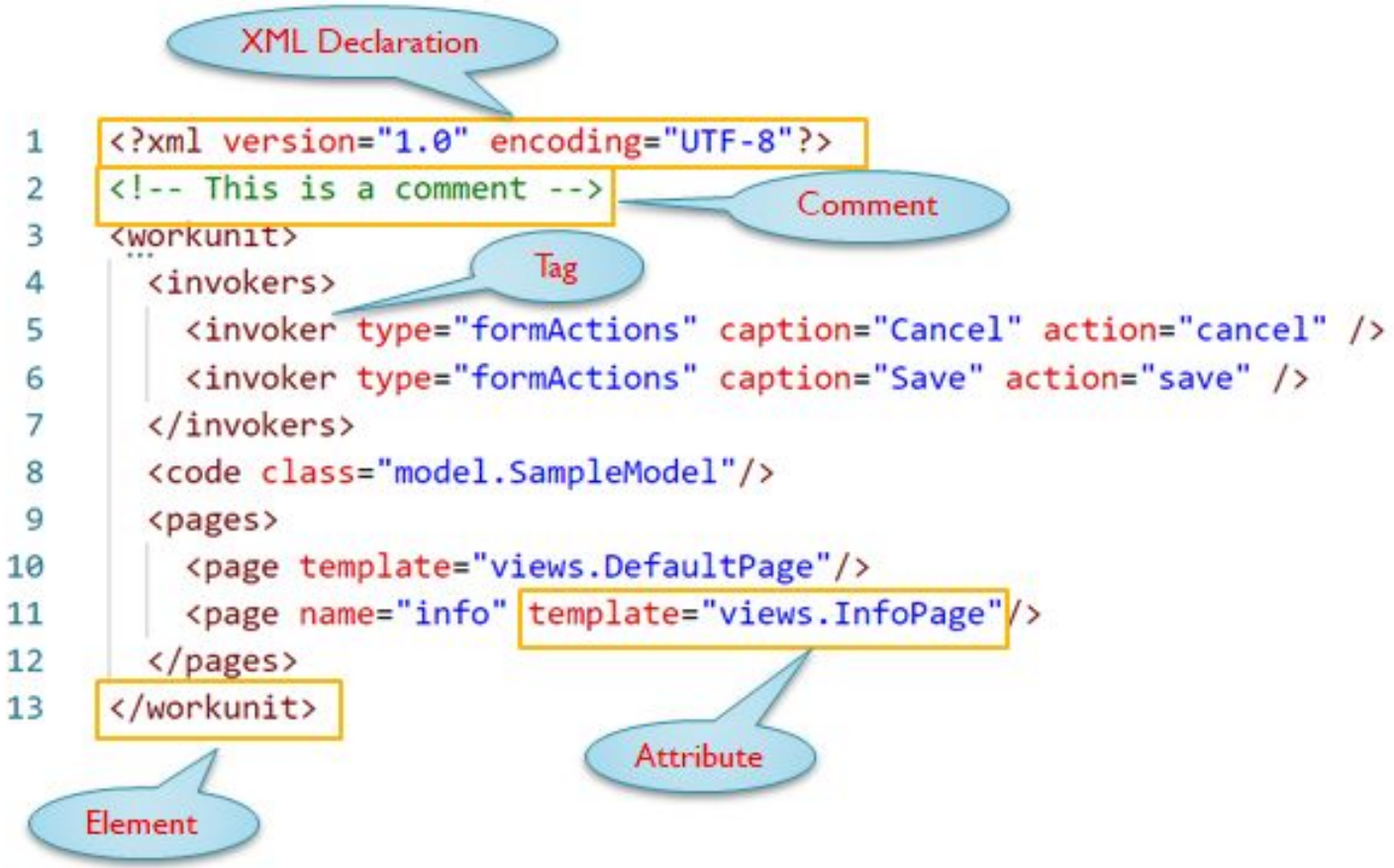
- Osiris3 Client Framework
- Basic Server Side Programming

XML LANGUAGE

XML Language

- A Markup language
- Self-descriptive
- Hierarchical structure
- Platform and language independent
- Readable and human-friendly.

XML Syntax


XML Declaration
Comment
Tag
Attribute
Element
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- This is a comment -->
3 <workunit>
4 <invokers>
5 <invoker type="formActions" caption="Cancel" action="cancel" />
6 <invoker type="formActions" caption="Save" action="save" />
7 </invokers>
8 <code class="model.SampleModel"/>
9 <pages>
10 <page template="views.DefaultPage"/>
11 <page name="info" template="views.InfoPage"/>
12 </pages>
13 </workunit>

XML Restrictions

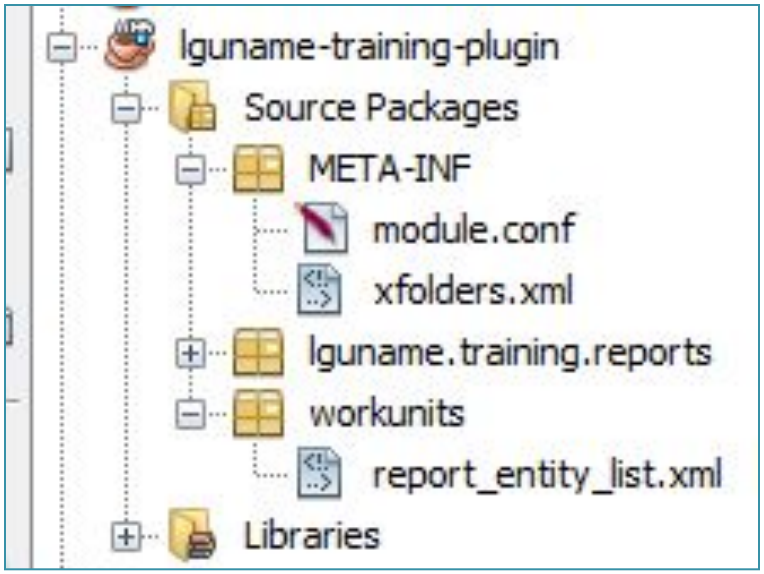
- Well-Formed
  - Proper Nesting
    - Every opening tag must have a corresponding closing tag
  - Tag Names
    - must start with letter or underscore
  - Attribute Values
    - Must be enclosed in quotes (single or double)

OSIRIS3 CLIENT FRAMEWORK

Osiris3 Client Platform Framework

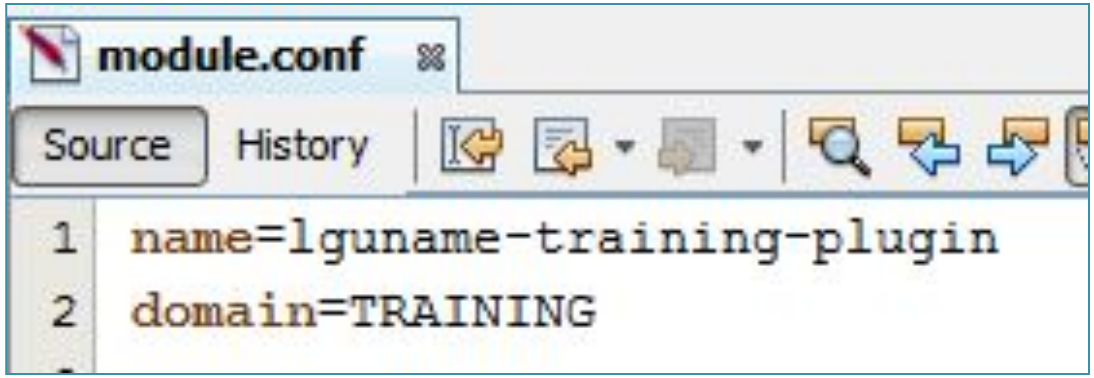
- Proprietary platform develop by Rameses
  - simplifies client development
  - uses Java or Groovy
  - proprietary UI Controls with binding support
  - uses MVC Pattern
  - pluggable capability
- Tight Integration with Osiris3 Server
  - easy services integration and development

ETRACS Plugin Module Structure



- **META-INF**
  - identifies the project as a plugin
  - **module.conf** – module information/configuration
  - **xfolders.xml** – defines entry point for a workunit
- **workunits**
  - contains workunit that represents a transaction, document, reports etc

The **META-INF**/module.conf File



- module configuration and information
  - **name**
    - the name of the module
  - **domain**
    - domain or group where the plugin is associated
    - reference in the security role

The **META-INF**/xfolders.xml File

xfolders.xml

Source | History

```
1  <folders>
2      <folder id="home">
3          <folder id="lguname" caption="LGU">
4              <folder id="reports" caption="Reports" />
5          </folder>
6      </folder>
7  </folders>
```

- An entry point for workunits
- Support any number of nested levels
- Types
  - **menu** – displays workunit in the Menu bar
  - **explorer** – displays workunit in Explorer window
  - **home** – displays workunit in the Home page
  - **menucategory** – displays workunit in the Home page in categorized format

The Folder Element

- Can be nested to any levels
- Key Attributes
  - **id** – the assigned ID included as part of the folder path
  - **caption** – the folder caption
  - **mnemonic** – assigned shortcut character
  - **icon** – assigned item icon

The **workunits** Folder

- contains controllers for managing user interface and service interaction
- holds one controller per document, either master files or transactions
- workunits are in **xml** format

The **workunit** Structure

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <workunit>
3      <invokers>
4      </invokers>
5      <code />
6      <pages>
7      </pages>
8  </workunit>
9
```

- **invokers** – defines access point for the controller
  - can either be a menu, home folder or via InvokerUtil
- **code** – the controller code either in Groovy or Java
  - it can either be inlined or external
- **pages** – the list of pages required by the controller

The **invoker** Element

- defines access points for the controller
  - can either be a menu, home folder or via InvokerUtil
- Key Attributes
  - **folderid** – menu item path where the invoker will be shown
  - **type** – a user defined type that can be used for lookup
  - **action** – the method executed when the invoker is called
  - **caption** – the caption for this invoker
  - **icon** – the assigned icon
  - **visibleWhen** – makes the invoker visible or hidden. The expression format is **#{ expr }**. If the expr is true, the invoker is visible.
    - Example : **visibleWhen="#{mode == 'init'}"**
  - **role** – the authorized role
  - **permission** – the authorized permission (per role if not set)
  - **mnemonic** – the assigned mnemonic character accelerator
  - **shortcut** – the assigned shortcut such as "ctrl S"

The **code** Element

- code can be external (Java or Groovy)
  - use the **class** attribute
  - Groovy is highly recommended
- code can be “inline”
  - part of the controller.xml file

The **pages** Element

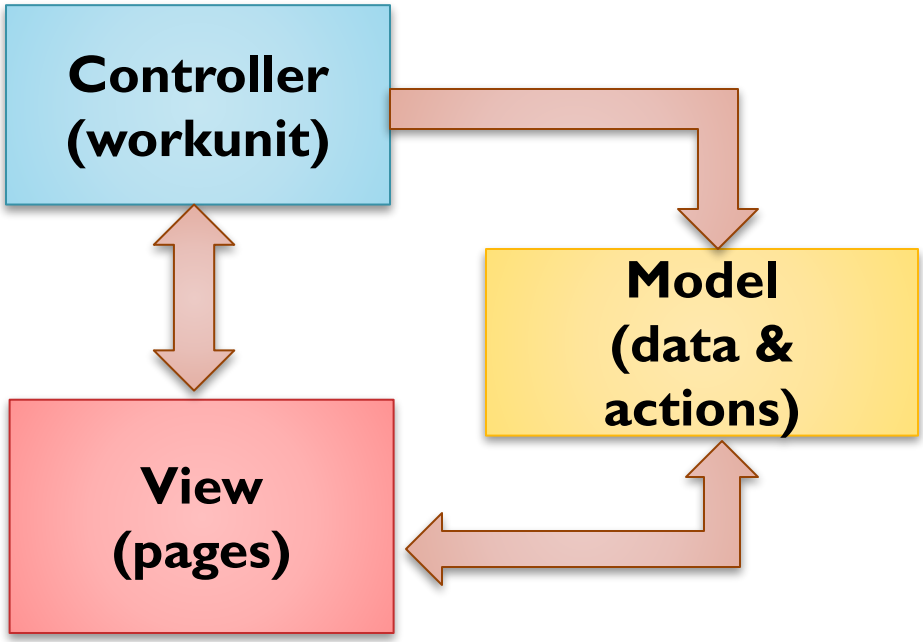
- list of pages required by the controller
- Key Attributes
  - **name** – the assigned name of the page
  - **template** – the complete path of the page class

NOTE:

"default" is the name specified if the name attribute is not defined.

The MVC Pattern

- Data binding support
- View actions are represented as methods in the model



The **osiris3** UI Controls

- Proprietary UI Controls
- Extended Swing UI with features such as data binding and simple validation
- Designed to simplify UI development

Basic UI Controls

- XFormPanel – container with improve layout capability
- XActionBar – holder of invoker actions
- XTextField – accepts text or string value
- XDecimalField – accepts decimal value
- XIntegerField – accepts integer value
- XLabel – displays static text
- XButton – represents a button
- XDateField – accepts date value
- XComboBox – represents a combo box
- XCheckBox – represents a check box

XFormPanel Control

- A container that organizes controls horizontally or vertically
- Key Properties
  - border
  - captionBorder
  - captionForeground
  - captionHAlignment
  - captionMnemonic
  - captionOrientation
  - captionPadding
  - captionVAlignment
  - captionWidth
  - cellPadding
  - padding

XActionBar Control

- A container that displays invoker actions
- Key Properties
  - **name** – then invoker type value to display
  - **useToolbar** – display as toolbar buttons
  - **visibleWhen** – expression to control visibility
  - **depends** – list of depended controls
  -

XTextField Control

- Accepts a string data
- Key Properties
  - **name** – the data binding name
  - **depends** – list of depended controls
  - **preferredSize** – the preferred size
  - **required** – set the field as required
  - **textCase** – set the string case
  - **showCaption** – flag to display/hide caption
  - **cellPadding** – adds padding spaces

XDecimalField Control

- Accepts a decimal input
- Key Properties
  - **name** – the data binding name
  - **depends** – list of depended controls
  - **preferredSize** – the preferred size
  - **required** – set the field as required
  - **cellPadding** – adds padding spaces

XIntegerField Control

- Accepts a integer input
- Key Properties
  - **name** – the data binding name
  - **depends** – list of depended controls
  - **preferredSize** – the preferred size
  - **required** – set the field as required
  - **cellPadding** – adds padding spaces

XLabel Control

- Displays a readonly information
- Key Properties
  - **name** – the data binding name
  - **depends** – list of depended controls
  - **preferredSize** – the preferred size
  - **showCaption** – flag to display/hide caption
  - **expression** – display the value of an expression
    - #{ expression}
  - **cellPadding** – adds padding spaces

XButton Control

- **name -** the method to be called
- **text** – the caption
- **depends** – list of depended controls
- **mnemonic**
- **shortcut**

INTRODUCTION TO OSIRIS3 SERVICES

Service Oriented Architecture

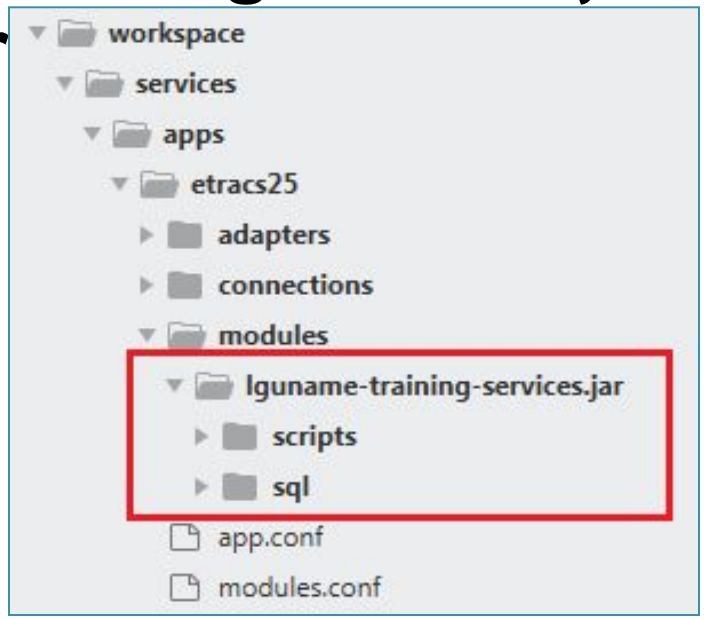
- a software architecture design pattern wherein distinct software provides application functionality as "services" to other applications.
- independent of any vendor, product or technology

Service Structure

- The service folder must end with **".jar"** extension
- It can contain the following folders:
  - **schema** – defines data/table definition
  - **scripts** – contains business logic codes
  - **sql** – contains SQL statements

The **lguname-training-services.jar**

1. Open in Windows Explorer the **{ETRACS_SERVER}/workspace/services/apps/etracs25/modules** folder
2. Create the folder **lguname-training-services.jar**. Add the folders **scr**



Service Annotations

What are Service Annotations?

- Framework specific settings that provide uniform way of accessing server resources
- Types of resources
  - Environment resources
  - Services
  - Databases
  - Methods
- Defined in the following imports
  - import com.rameses.annotations.*
  - import com.rameses.common.*
  - import com.rameses.services.extended.*

@Env Annotation

- use to access the current Environment
- access information such as
  - USERID, USERNAME, USER, FULLNAME
  - ORGID, ORGCODE, ORGCLASS

- Example declaration

```
@Env
def env
```

@Service Annotation

- use to reference a service name
- once a service is successfully referenced, service methods can be called and executed
- available within a Service or Osiris Client
- Syntax
  - @Service(service_name)
- Example Declaration

```
@Service('PersonnelService')
def personnelSvc
```

@ActiveDB Annotation

- use to access a particular **schema** or **sql**
- for ActiveDB with associated **schema**, the following methods are available:
  - **create -** to insert table data
  - **read** – to read table data
  - **update** – to update table data
  - **delete** – to delete table data
  - **save** – to insert or update table data
- for ActiveDB with associated **sql**, statements are directly called through its **[sql_name]**

@ActiveDB Annotation

- Syntax
  - **@ActiveDB(schema_or_sql_name)**
  - **@ActiveDB(value=schema_or_sql_name, em=adaptername)**

- Example Declaration

```
@ActiveDB('personnel)
def em;

@ActiveDB(value='personnel', em='training')
def em;
```

@ProxyMethod Annotation

- indicates that a class method is accessible or callable from other class or service
- the annotated method must be declared as **public**.

- Example Declaration

```
@ProxyMethod
public Map create(entity){
    return entity
}
```

The **entity** Variable

- Special variable name recognized by the platform
- The **entity** variable is passed from one invoker to another
- Allows the passing of data from model to another
- Should be used as the "**data**" variable for every model

End of Module