Policy Center

Lesson Outline

- Gosu Class
  - Create a Class
  - Extend a Class
- Logging & Debugging
- Exception Handling

Gosu Class

Guidewire Gosu

- Gosu is Guidewire's open-source, publicly available programming language
  - Has elements of both procedural and object-oriented programming languages
  - Similar to JavaScript and Java

- Gosu specifies runtime business logic that:
  - Executes fundamental application behavior
  - Manages complex business processes
  - Specifies dynamic client-side behavior

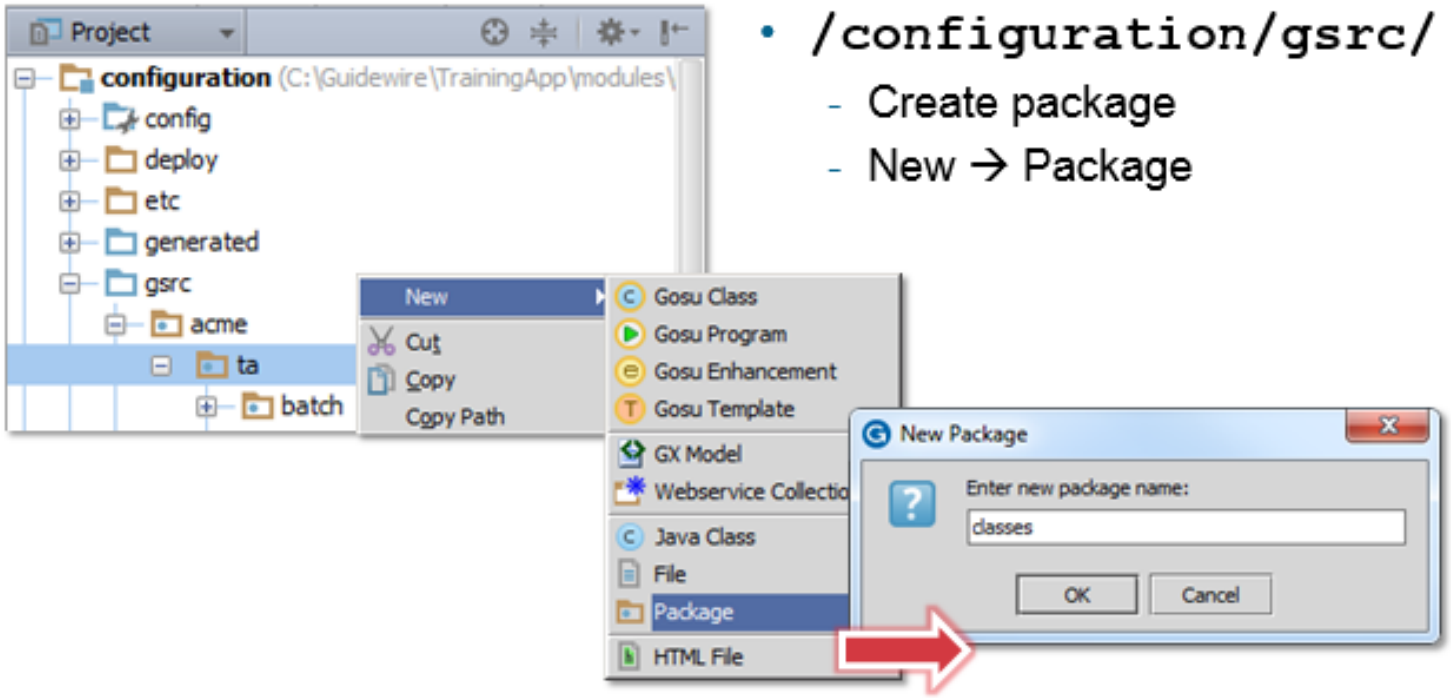
Gosu features and benefits

- Code is more compact
  - Direct access to Guidewire data model
- Development time is shorter
  - Reload Classes
- Guidewire Studio supports Gosu coding
  - Code completion
  - Syntax checking
  - Navigable links
- Robust interaction with Java
  - Easy to learn by Java programmers
  - Has access to all Java types and can reference Java classes

Gosu class features

- Gosu classes are similar to classes in other object-oriented language, such as Java
- Classes are organized in packages
- Classes can define:
  - Constructors
  - Properties (with private and public access)
  - Methods (with private and public access)
- Classes can:
  - Extend other classes
  - Implement interfaces
  - Override methods

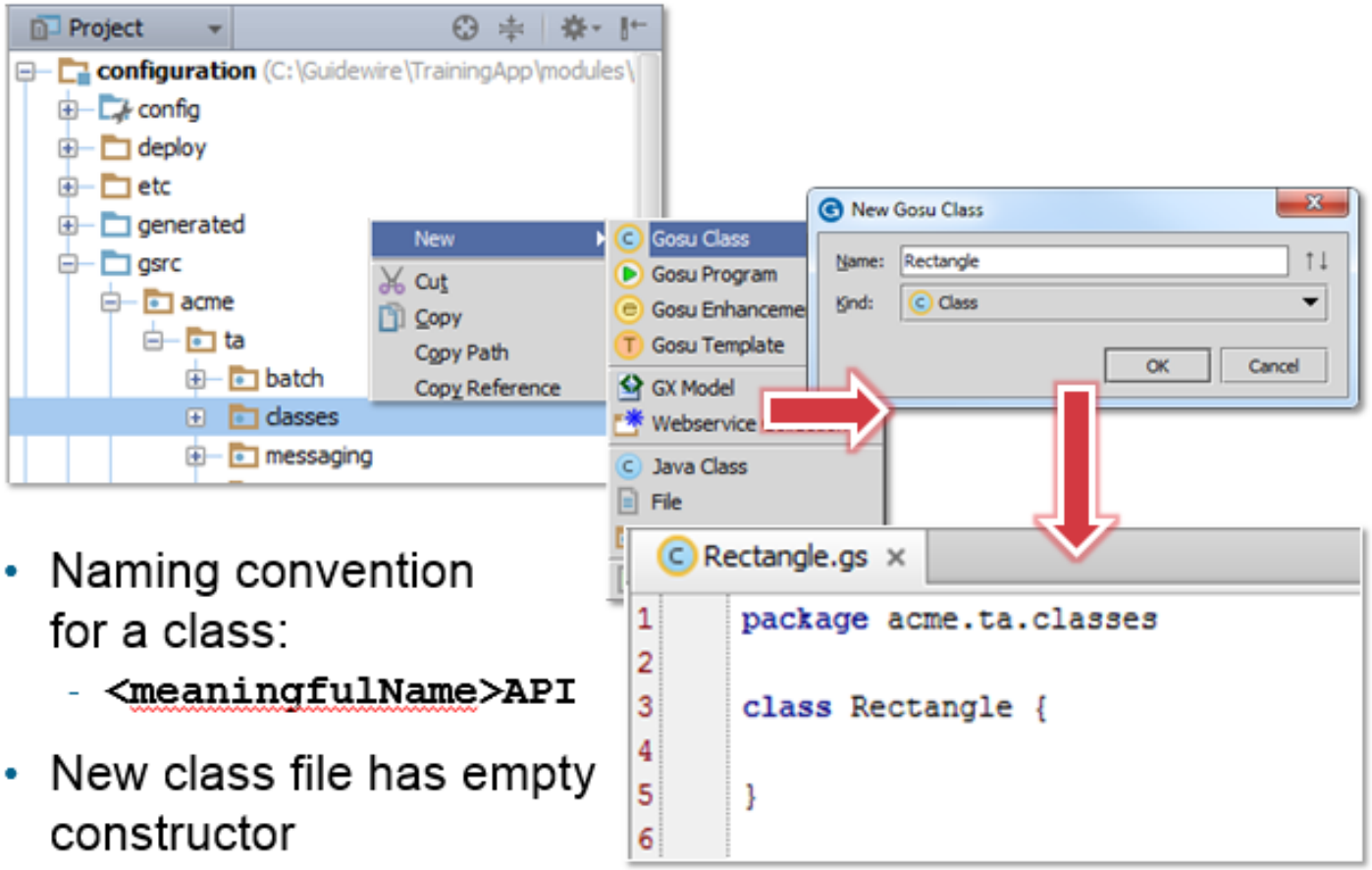
Creating Gosu packages



- `/configuration/gsrc/`
  - Create package
  - New → Package

- Guidewire naming convention recommendation
  - `<company>.<app_code>.mechanism.<functional_area>`
  - `mechanism` is typically batch, messaging, plugin, startable, webservice, and of none of the others apply, then use class

Creating Gosu classes



- Naming convention for a class:
  - **<meaningfulName>API**
- New class file has empty constructor

Extending a class

Square.gs

```
5   class Square extends Rectangle implements ILineAndFill  {
6
7     construct() {
8       super()
9     }
10
26    ...
29    override function calculateArea() : int {
30      // This overrides Rectangle's
31      // calculateArea() method
32      return (Width * Width)
33    }
34
35    // Method declared in the Square class.
36    // It is not inherited from anywhere.
37
38    function calculatePerimeter() : int {
39      // This method is unique to
40      // Square class
41      return (4 * Width)
42    }
```

Access to inherited properties

- To extend a class
  - add extends superClass to the class declaration
- Super keyword gives you access to aspects of the superclass
- Override keyword lets you override methods declared in the super class

- As declared in the superclass, the class can access non-private properties and methods

Logging and Debugging

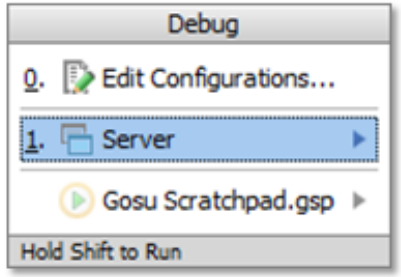
Logging

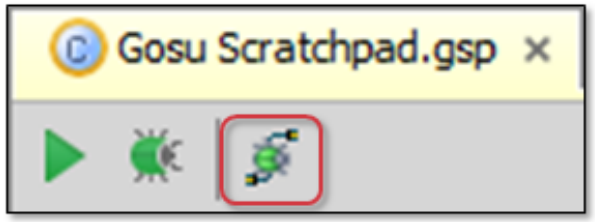
```
35  function enlarge(factor: int): String {
36    if (factor <= 0) {
37      gw.api.util.Logger.logInfo("Failed attempt to enlarge by an invalid factor")
38      throw new gw.api.util.DisplayableException("Requires a factor greater than 0")
39    }
40    var originalArea = calculateArea()
41    _width = _width * factor
42    _height = _height * factor
43    return "Area of rectangle increased from " + originalArea +
44        " to " + calculateArea()
45  }
```

- Write to the log using the logger class
  - **`gw.api.util.Logger.logTypeXX("logString")`**
  - *TypeXXX* is for
    - Trace, Debug, Info, Warn,
    - Error

Steps to run in a debug process

1. Debug Server
   - **Alt + Shift + F9**
   - Select Server
2. Console tab
   - Verify output reads application ready
3. Open Gosu scratchpad
   - **Alt + Shift + S**
4. Enter Gosu code in scratchpad
   - Able to reference project entities, classes, libraries, and SDK
5. Run in Debug Process
   - No connection dialog!





Exception Handling

```
35  function enlarge(factor: int): String {
36    if (factor <= 0) {
37      gw.api.util.Logger.logInfo("Failed attempt to enlarge by an invalid factor")
38      throw new gw.api.util.DisplayableException("Requires a factor greater than 0")
39    }
40    var originalArea = calculateArea()
41    _width = _width * factor
42    _height = _height * factor
43    return "Area of rectangle increased from " + originalArea +
44        " to " + calculateArea()
45  }
```

- Gosu can make use of **`try`**...**`catch`**...**`finally`** blocks
- Gosu code can throw Gosu and Java exceptions

Task

- Functionality to calculate Take Home salary from Cost to Company.
- Create a class TaxInfo with function taxesOnSalary() to calculate sum of taxes.
- Create a class SalaryInfo , extends TaxInfo and add function to calculate take home salary after deducting taxes.
- Add loggers and handle exceptions

Summary

- Gosu Class
  - Create a Class
  - Extend a Class
- Logging & Debugging
- Exception Handling

Thank You