Adventures in TclOO

or, Here's Some Tricks I've Been Up To

Donal Fellows
University of Manchester / Tcl Core Team

Outline

* A Quick TclOO Refresher
* Support Class for REST
* Wrapping TDBC with ORM
* Extending TclOO with Annotations
* Where to go in the Future

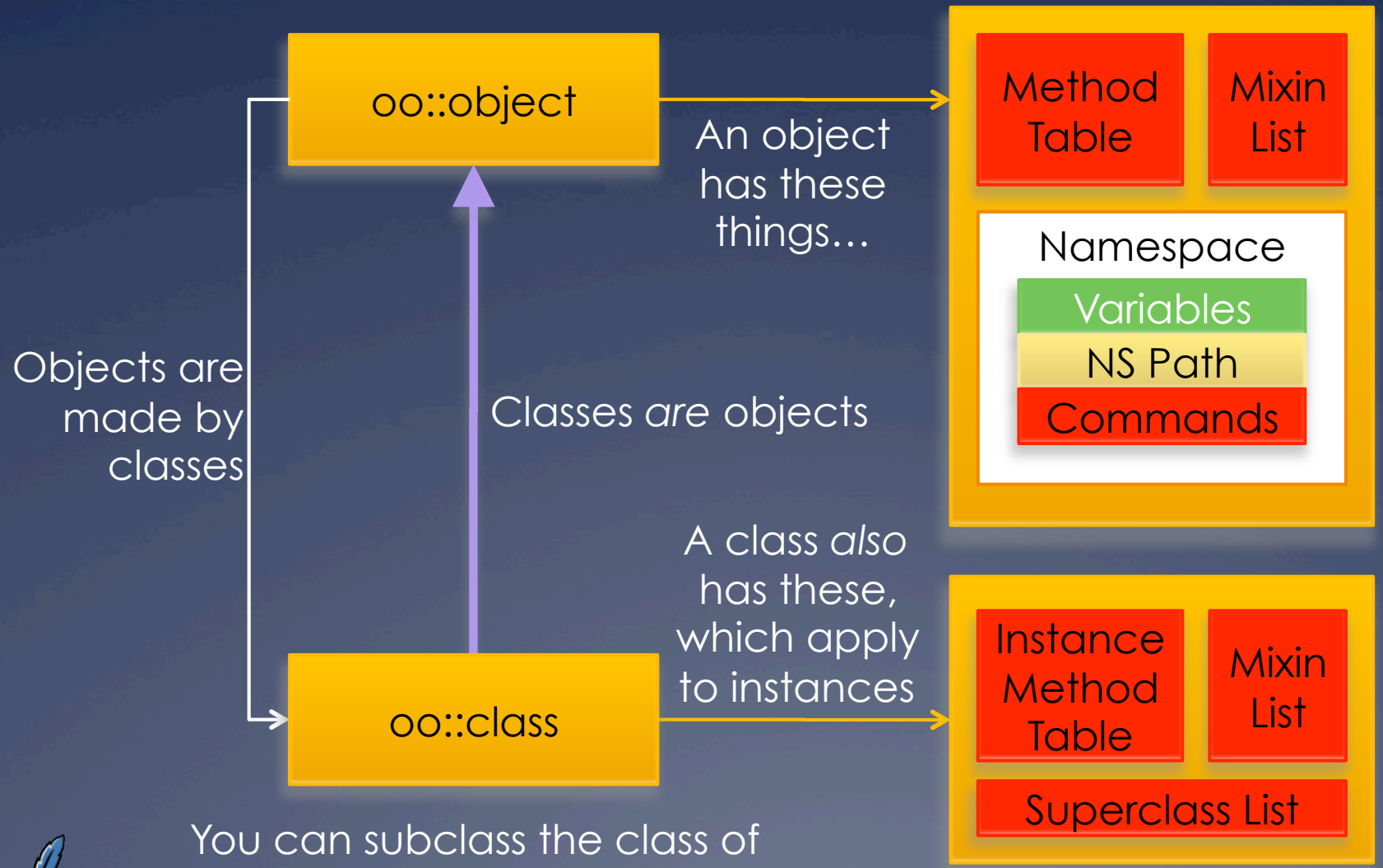
A Quick TclOO…

Refresher

TclOO Refresher

- Object System
  - Incorporated in 8.6, Package for 8.5
- Designed for
  - Keeping Out of Your Way
  - Being Tcl'ish and Dynamic
  - Being Powerful and Extensible
- Key Features
  - Single-Rooted Inheritance Hierarchy
  - Classes are Objects and May be Extended
  - Object's Namespace contains Object's Variables
  - Class (Re-)Definition by Separate Command

The Anatomy Lesson



Support Class for…

REST

REpresentational State Transfer

- REST is Way of Doing Web Services
  - Popular
  - Easy to Use in *Ad Hoc* Way
- Strongly Leverages HTTP
  - Verbs are GET, PUT, DELETE, POST, …
  - Nouns are Resources with URLs
  - View of Resources Determined by Content Negotiation

Example…

- **http://example.org/pizzas** is a Pizza Parlor
- **GET http://example.org/pizzas**
  - Returns Current List of Pizzas
- **GET http://example.org/pizzas/123**
  - Returns Description of Pizza #123
- **GET http://example.org/pizzas/123/pepperoni**
  - Returns Amount of Pepperoni on Pizza

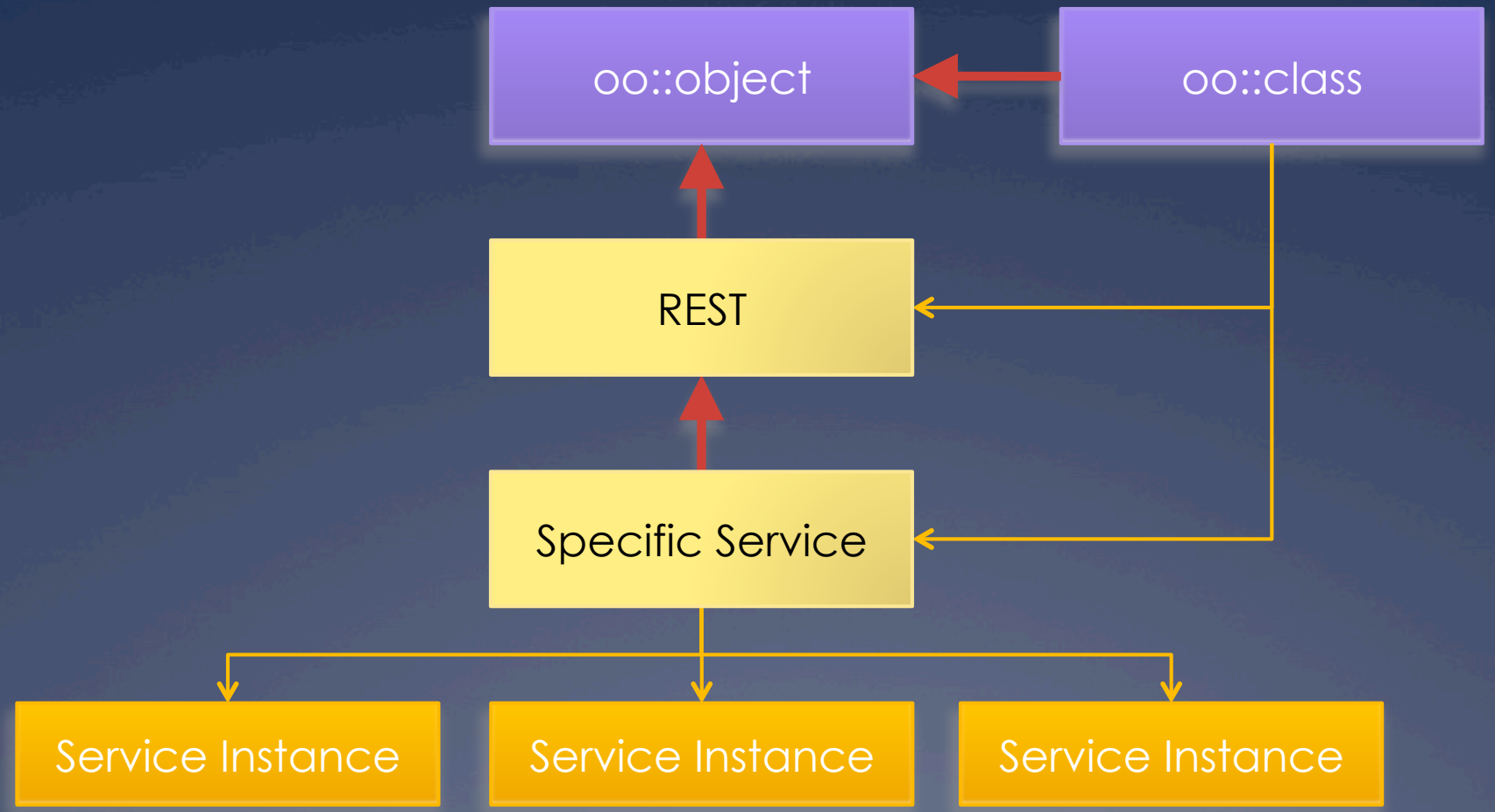
Example… Updates

- **PUT http://example.org/pizzas/123/pepperoni**
  - Sets Amount of Pepperoni on Pizza #123
  - Idempotent
- **POST http://example.org/pizzas**
  - Makes a New Pizza from Scratch
    - Request document says what to make
  - Redirects to Created Pizza
- **DELETE http://example.org/pizzas/123**
  - Gets Rid of Pizza #123
  - Idempotent

REST Class

- TclOO Class
  - Wrapper for http package
  - Models a Base Resource
  - Methods for Each HTTP Verb
  - Designed to be Subclassed!
- Some Production Use
  - Testing Interface for Very Complex REST Service
  - More than 35 Operations with non-fixed URLs
    - Still Growing…

General Plan



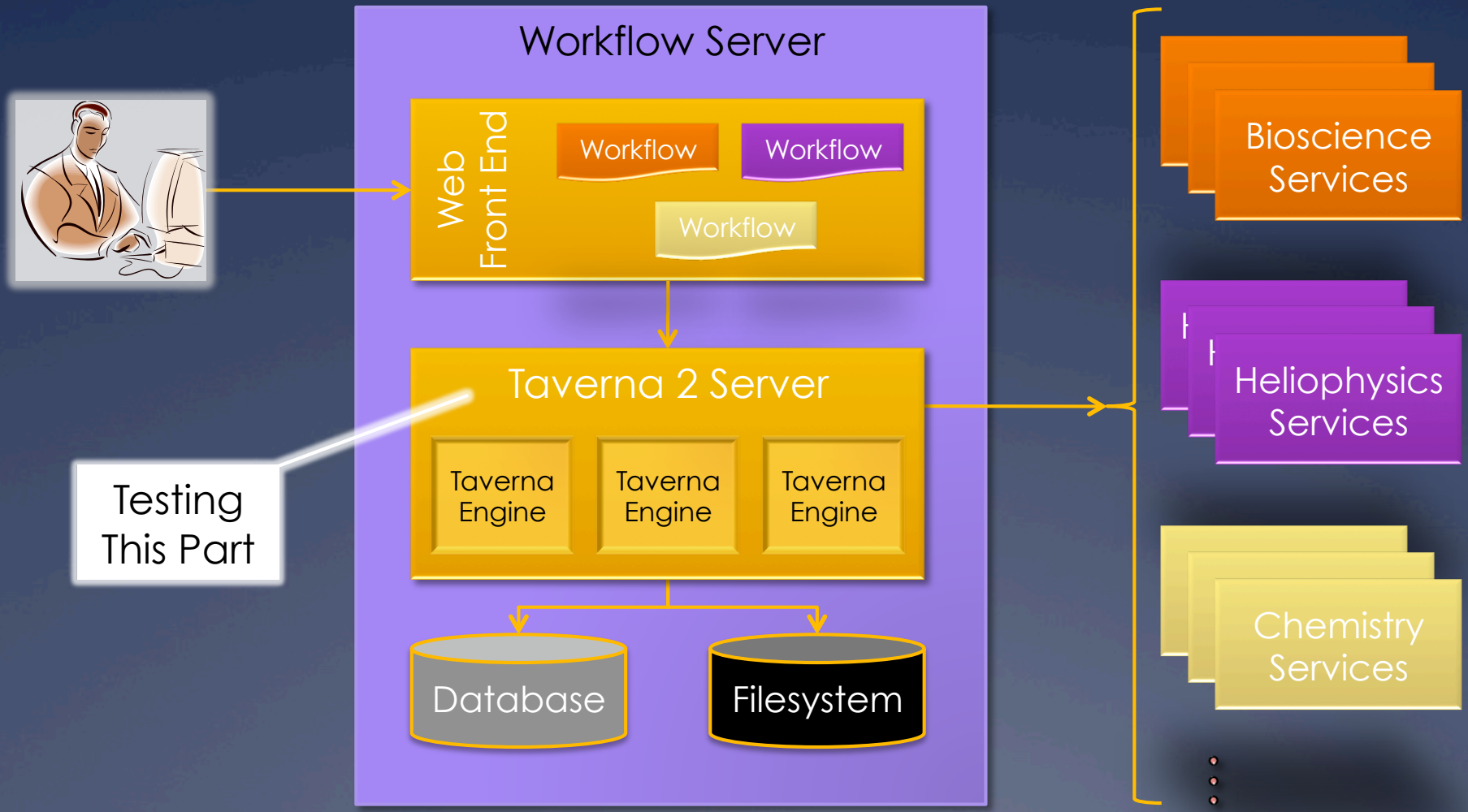
Code (Simplified)

```
method DoRequest {
    method url {type ""} {value ""}} {

  for {set rs 0} {$rs < 5} {incr rs} {
    set tok [http::geturl $url \
        -method $method \
        -type $type -query $value]

    try {
      if {[http::ncode $tok] > 399} {
        # ERROR
        set msg [my ExtractError $tok]
        return -code error $msg

      } elseif {[http::ncode $tok] > 299
          || [http::ncode $tok]==201} {
        # REDIRECT
        try {
          set location [dict get \
              [http::meta $tok] Location]
        } on error {} {
          error "missing location!"
        }
        my OnRedirect $tok $location

      } else {
        # SUCCESS
        return [http::data $tok]
      }
    } finally {
      http::cleanup $tok
    }
  }
  error "too many redirections!"
}


method GET args {
  return [my DoRequest GET \
      $base/[join $args "/"]]
}
```

Usage

```
oo::class create Service {
    superclass REST
    # ... etc ...

    method status {{status ""}} {
        if {$status eq ""} {
            return [my GET status]
        }
        my PUT status text/plain $status
        return
    }
}
```

What I Was Using This For



Wrapping TDBC with…

ORM

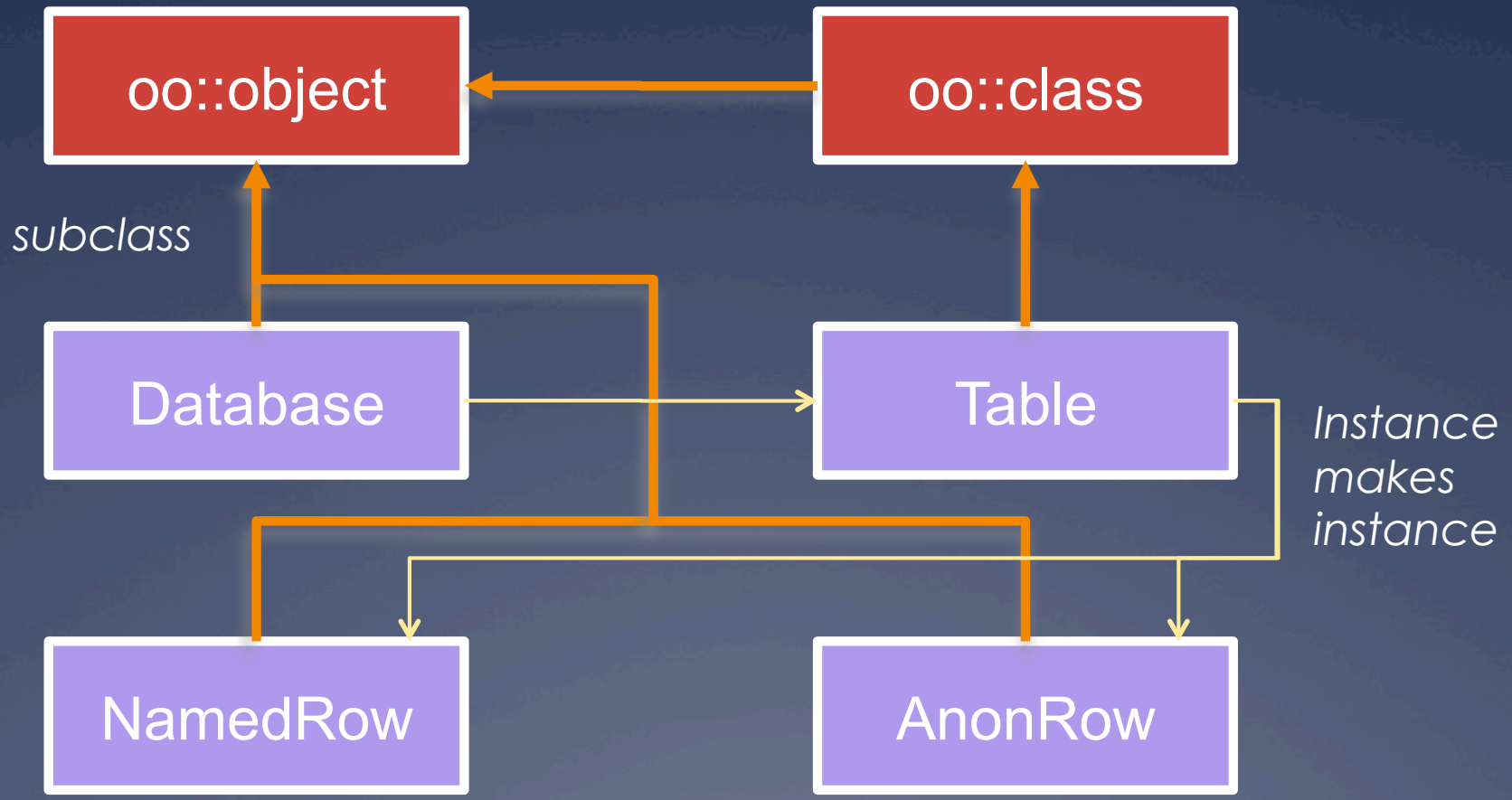
Object-Relational Mapping

- Objects are Natural Way to Model World
  - Well, much of it…
- Relational Databases are Excellent at Managing Data
- Build Links between Objects and Databases
  - Many Languages Have Them
    - Java, C#, Ruby, …
  - Wanted to do in Tcl
    - Leverage TclOO and TDBC

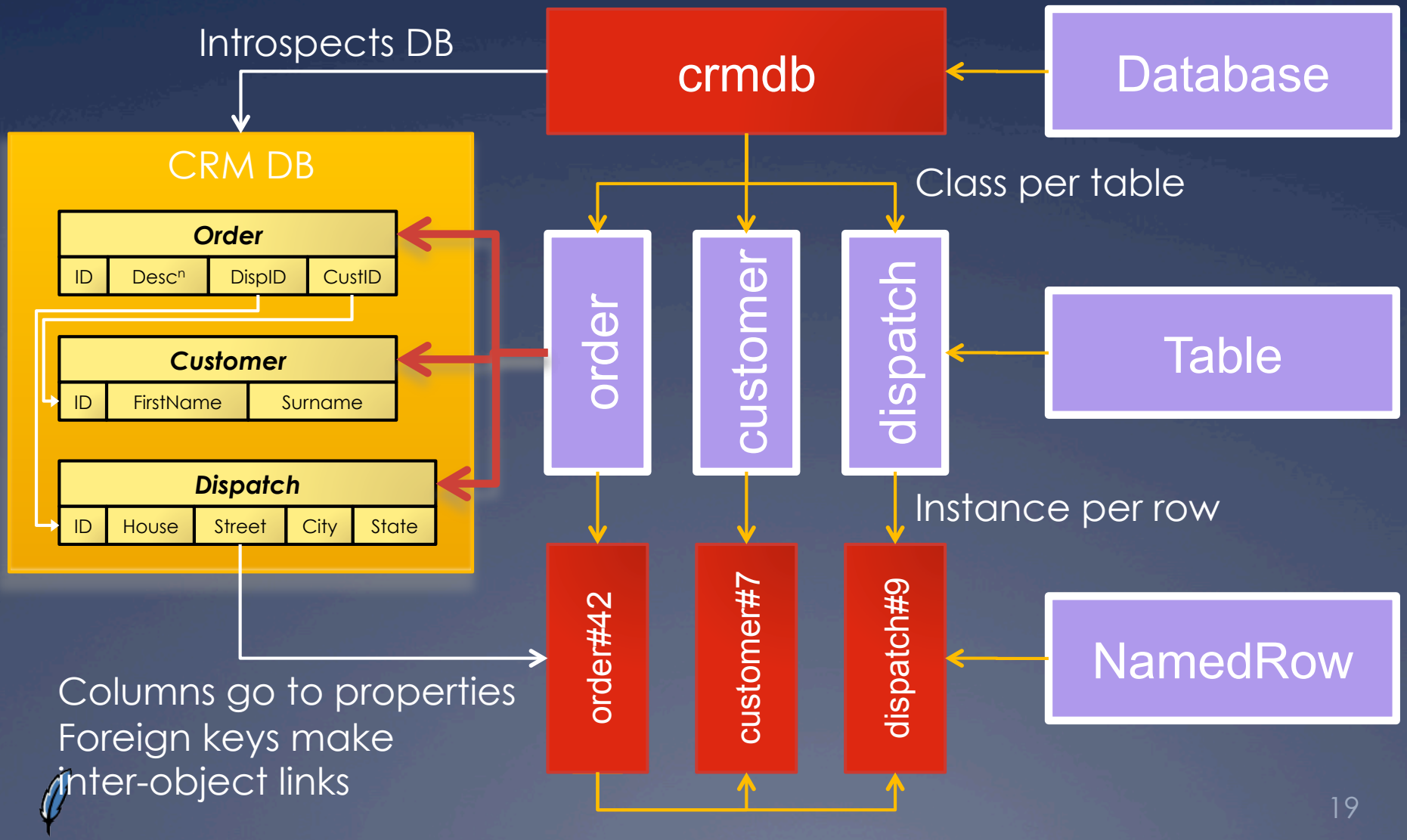
Data First or Objects First?

- Data First
  - Data is already in the database
  - How to introspect database's structure
  - How to represent naturally as objects
- Objects First
  - Data is in objects
  - How to construct database to store and restore
- My ORM Package is Data First
  - Dynamic class construction!

Basic Class Plan



Interaction Plan



Example of Use

```
# Connect to Database with TDBC
set conn [tdbc::sqlite3::connection new "mydb.sqlite3"]

# Create all the classes holding the mapping
ORM::Database create db $conn

# Illustrate use by printing out all orders
db table order foreach ordr {
    puts "Order #[$ordr id]"
    puts "Customer: [[$ordr customer] firstname]\
          [[$ordr customer] surname]"
    puts "Address: [[$ordr dispatch] house]\
          [[$ordr dispatch] street]"
    puts "Address: [[$ordr dispatch] city],\
          [[$ordr dispatch] state]"
    puts "Description:\n\t[$ordr description]"
    puts ""
}
```

Extending TclOO with…

Annotations

What is an Annotation?

- Additional Arbitrary Metadata
  - Attached to Class or Part of Class
  - User-Defined Purpose
    - Easy way to add information without doing big changes to TclOO's C implementation
- Uses from Other Languages
  - Documentation
  - Constraints
  - Coupling to Container Frameworks
    - Persistence, Web Apps, Management, …

Annotation Syntax

```
@SomeAnnotation -foo bar
oo::class create Example {

    @AnotherAnnotation
    variable x

    @GuessWhatThisIs
    constructor {} { … }

    @HowAboutThis?
    @More…
    method xyzzy {x y} { … }
}

# To read the annotations…
puts [info class annotation Example]
```

Types of Annotations

- Annotations Specify What They Attach To
  - Classes
  - Methods
  - Any declaration…
- Examples
  - @Description
  - @Argument
  - @Result
  - @SideEffect

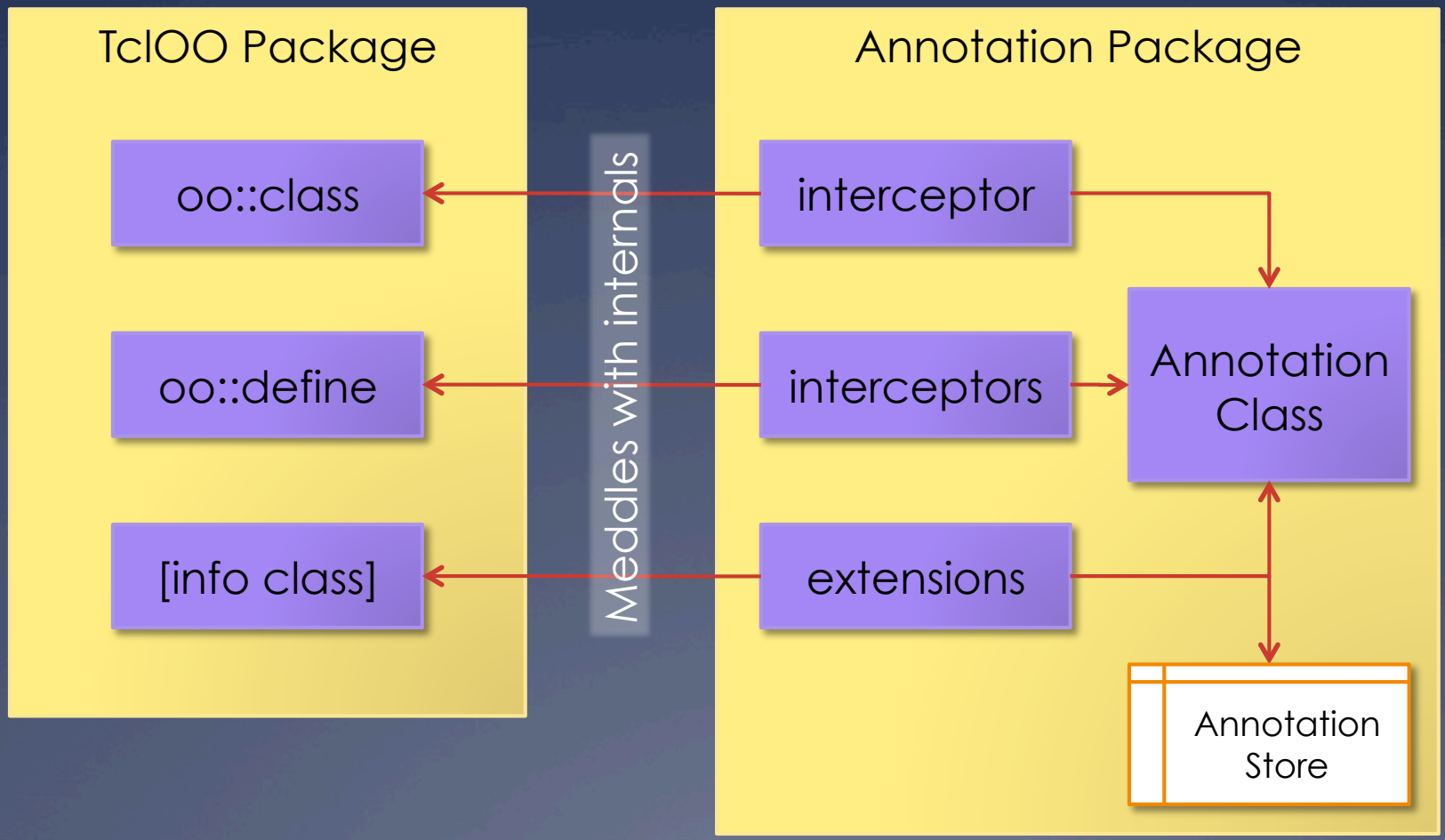
Declaring an Annotation

```
oo::class create Annotation.Argument {
  superclass Annotation.Describe
  variable annotation method \
      argument

  # Save what argument this applies to
  constructor {type argName args} {
    set argument $argName
    next $type {*}$args
  }

  # How an annotation is introspected
  method describe {v {n ""} {an ""}} {
    upvar 1 $v result
    if {[llength [info level 0]] == 3} {
      lappend result $method
      return
    }
    if {$method ne $n} {
      return
    }
    if {[llength [info level 0]] == 4} {
      lappend result $argument
      return
    } elseif {$argument eq $a} {
      set result [join $annotation]
      return -code break
    }
  }
}
```

Under the Hood

- Every Class has List of Annotations Applied to it
- Attaching to Classes
  - Uses Standard Unknown Handler
  - Rewrites oo::class Command
- Attaching to Declarations
  - Rewrites Whole TclOO Declaration System
    - Change Declaration
    - Local Unknown Handler
- Adding Introspection
  - Extra commands in [info class] via ensemble

Under the Hood



Where to go in the

Future?

Future of REST Class

- Contribute to Tcllib
- Needs More Features First
  - Authentication Support
  - Cookie Handling
  - WADL Parser
    - Well, maybe…

Future of ORM

- Much Work Needed
  - What is natural way to handle object deletion?
  - What is best way to handle complex keys?
  - What is best way to bring in objects?
  - What sort of cache policy should be used?
- Support Object-First Style
  - With annotations?

Future of Annotations

- Too Hard to Declare
  - Far too much work!
- Introspection Axes Wrong
  - Current code uses very unnatural order
- Add to TclOO?
  - Depends on issues being resolved
- Find Cool Things to Do, e.g…
  - Augment ORM?
  - Ways of *creating* server-side REST bindings