Welcome

To Advance through Presentation
Use Page Up and Page Down Keys

WebObjects

"This year's Software Product of the Year is Apple Computer's WebObjects 4.0. . .Along with substantial power and flexibility, WebObjects brings an ease of development rarely seen in industrial-strength development environments."

WebObjects Debugging Techniques

Mai Nguyen,
Developer Support
Bob Frank,
Consulting Engineer
Steve Hayman,
Systems Engineer

Debugging Road Map

- Proactive Debugging
  (Don't write bugs—doh!)
- Deep in the Trenches
  (OK, well now what?)
- Debugging Contest
- Q & A

Avoid Bugs, Be Proactive!

- Would you rather:
  - Write new code?
  - Fix 9-month old code?
- Use standard coding conventions
- Compile frequently
- Use SCM

Coding Conventions

- Capitalization

  classes: MyClass (ex. NSObject, WOComponent)

  methods: myMethod (ex. [localObject aMessage],
  [NSString stringWithFormat: @"Hi there world"])

  Java method: localObject.aMessage();

  iVars: myAttribute, myVariable

Coding Conventions

- Return values!
- Make it hard to ignore error conditions—use Exceptions (e.g., next slide)
- Don’t combine error conditions with return values

Use Exceptions

- In Objective-C:

```
NS_DURING
    [editingContext saveChanges];
NS_HANDLER
    [NSException raise:  NSGenericException
                withFormat: @”Debug info: %@", [iVar
method]];
NS_ENDHANDLER
```

- In Java:

```
try {
    // Save all changes in your object graph into the
database
    editingContext.saveChanges();
}
catch (Throwable exception) {
    // go to a new error page or handle exception
}
```

Compile Frequently

- Catches typos and lazy syntax errors
- Use warnings—try to compile your projects without any warnings
- Use_debug libraries

A Warning

```
{  int x;
     switch (y) {
        case 1: x = 1; break;
        case 2: x = 4; break;
        case 3: x = 5; }
     foo (x);
}
```

warning: `x' might be used uninitialized in this function

- Uninitalized variables
- Also always use a “default” case
- = vs. ==

Using #warning

- It is a pre-processor command that will produce warning messages
- Use it to flag incomplete methods / classes / categories, if you are:
  - waiting on other code
  - waiting on application specifications
- Don’t use, if you can write the code
- Make sure that whatever is incomplete will not cause any bugs

Use SCM!

- CVS (Optimistic locking)
- PVCS, VSS (MS) (Pessimistic locking)
- VSS (MetroWerks), VooDoo (unknown locking strategies)

General Advice

- If it just stops working, what was the last thing you changed?
- When debugging, don't worry about performance. . . too much
- If your program doesn't run, then its performance sucks
- Less code is usually (better) code

More Naming Issues

- Know the key/value coding protocol!
- Standard accessor methods:
  setIVarName / getIVarName / iVarName

Spelling Issues

- Misspelled method names—
  They Won't Be Called!
- Misspelled keys in the key/value coding protocol—Raises Exceptions!

A Very Few Pointer Issues

- Adding nil to an array
- Inserting nil into a dictionary
- Creating a variable without allocating any memory (a classic)
- Trying to dereference a nil pointer

Super and id

- Overriding a method without calling super
- Avoid using “type” id
- Why?
    - Makes code more readable
    - Gives the compiler hints and allows better type checking at compile time
- Why not?
    - rapid proto-typing
    - temporary and throw-away variables

Misunderstanding WebObject's Framework Control Flow

- Understand the Request-Response loop
- awake vs. appendToResponse

Deep in the Trenches

- Prepare for debugging
- Tips for debugging WebObjects
- Tips for debugging EOF
- How about Java Web Apps?

Prepare for Debugging

- Use Project Builder to build your applications with the debug target
- To debug java applications, you also need to add in your Makefile.preamble:

```
OTHER_JAVATOOLS_FLAG = -g
```

or

```
OTHER_JAVAC_FLAGS = -g
```

Avoid Runtime Confusion

- Multiple projects with the same name
  - Understand **NSProjectSearchPath**
- Multiple build target binaries
- You must rebuild/restart your apps for changes in EOModel files, or code
  - If you touch a **.h** file, you must touch the **.m** file to pick up the changes

Gdb or jdb?

- Project Builder lets you switch between debuggers gdb and jdb
  - Limitation: no mixed stack trace
- Know differences between gdb and jdb:
  - You can't execute methods in jdb
  - Gdb stops all threads until you continue, jdb continues to give processor time to other threads

Gdb or jdb? (Cont.)

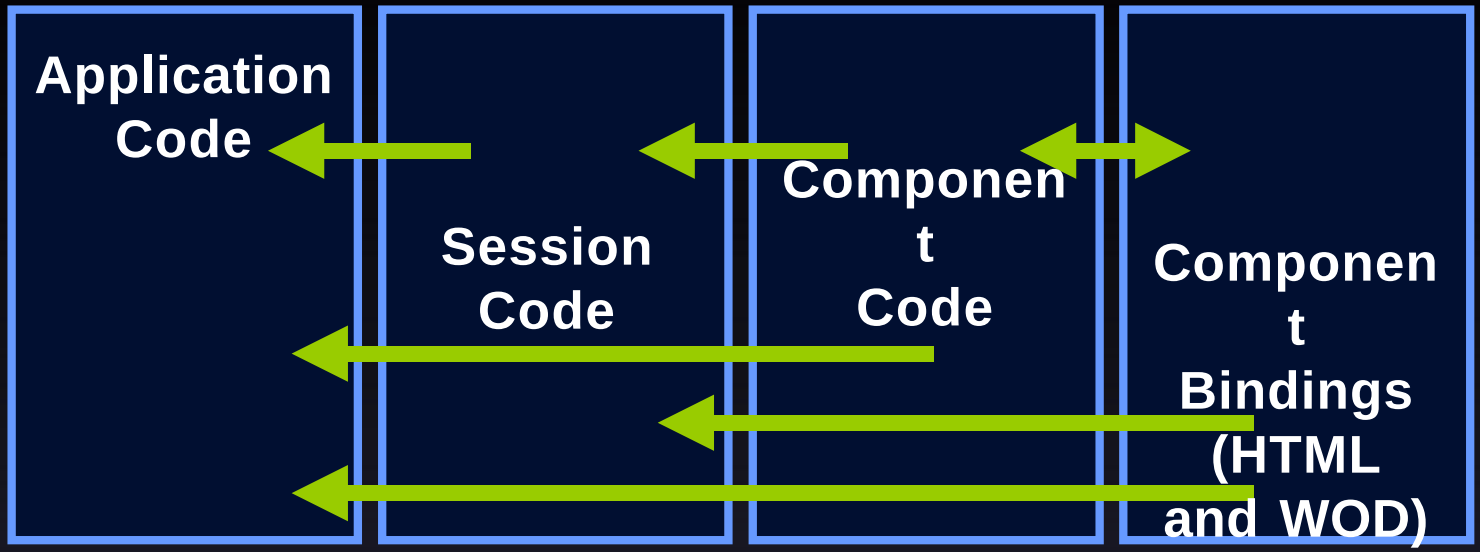
- Store often used gdb commands in a startup file (see sample gdb.ini for NT)

Demo

Project Builder,
gdb, jdb, Debug in
Mixed Language Environment

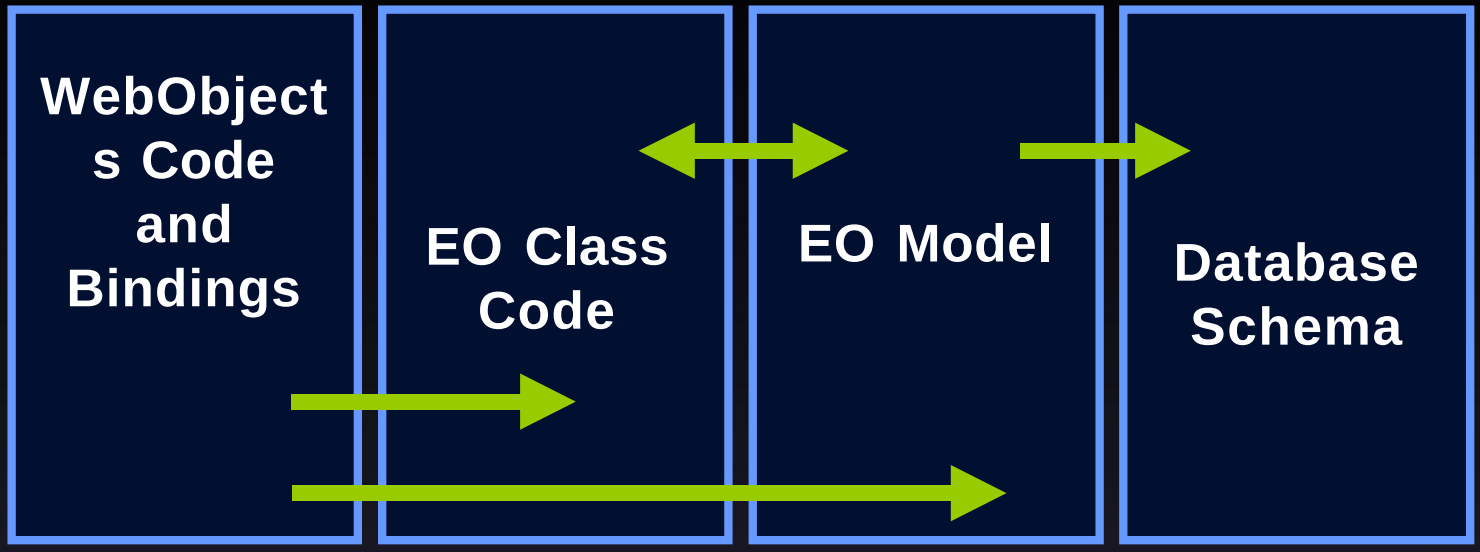
Components of a Web Application (1)

- WebObjects dependencies



Components of a Web Application (2)

- EOF Dependencies



Tips on Debugging WebObjects

- Use environment variables
  - WODebuggingEnabled
- Use tracing methods
  - WebScript, Objective C, Java
- Debug with or without the server
  - Direct Connect mode

Test with Different Web Browsers

- Your app can behave differently in different browsers
- View HTML source with the browser
- Turn Java Console on with the browser

Common WebObjects Binding Pitfalls

- Review the bindings of your components when encountering exceptions
    - Binding can be missing or have the wrong value
- Common bug: Use a literal value like “lastName” (quotes included) instead of a key name like lastName (no quotes)

WebObjects and HTML Pitfalls

- Multiple submit buttons must have multipleSubmit binding set to 1
- Locate nesting problems by viewing HTML source with WebObjects Builder

Tips on Debugging EOF

- Some useful environment variables:
  - EOAdaptorDebugEnabled
    - Logs connection attempts, all transaction activity, SQL statements.
  - EOFDebugEditingContext
    - EOF Control layer will log everytime an object is changed
  - EOFDebugUndo
    - Logs each time something is pushed/popped from the undo list

Tips on Debugging EOF

- Check your model file with EOModeler Consistency Check
    - When seeing problems with the database
- Use delegate methods or
- Use posers/categories (Objective C) to help debugging

Use Delegate Methods

- To check the control flow
  - **willDoSomeAction, didDoSomeAction**
  - **editingContextWillSaveChanges,**
- To handle error situations
  - **databaseContext:failedToFetchObject:globalID:**

Use Categories or Posers (Objective C)

- To add custom behavior for debugging

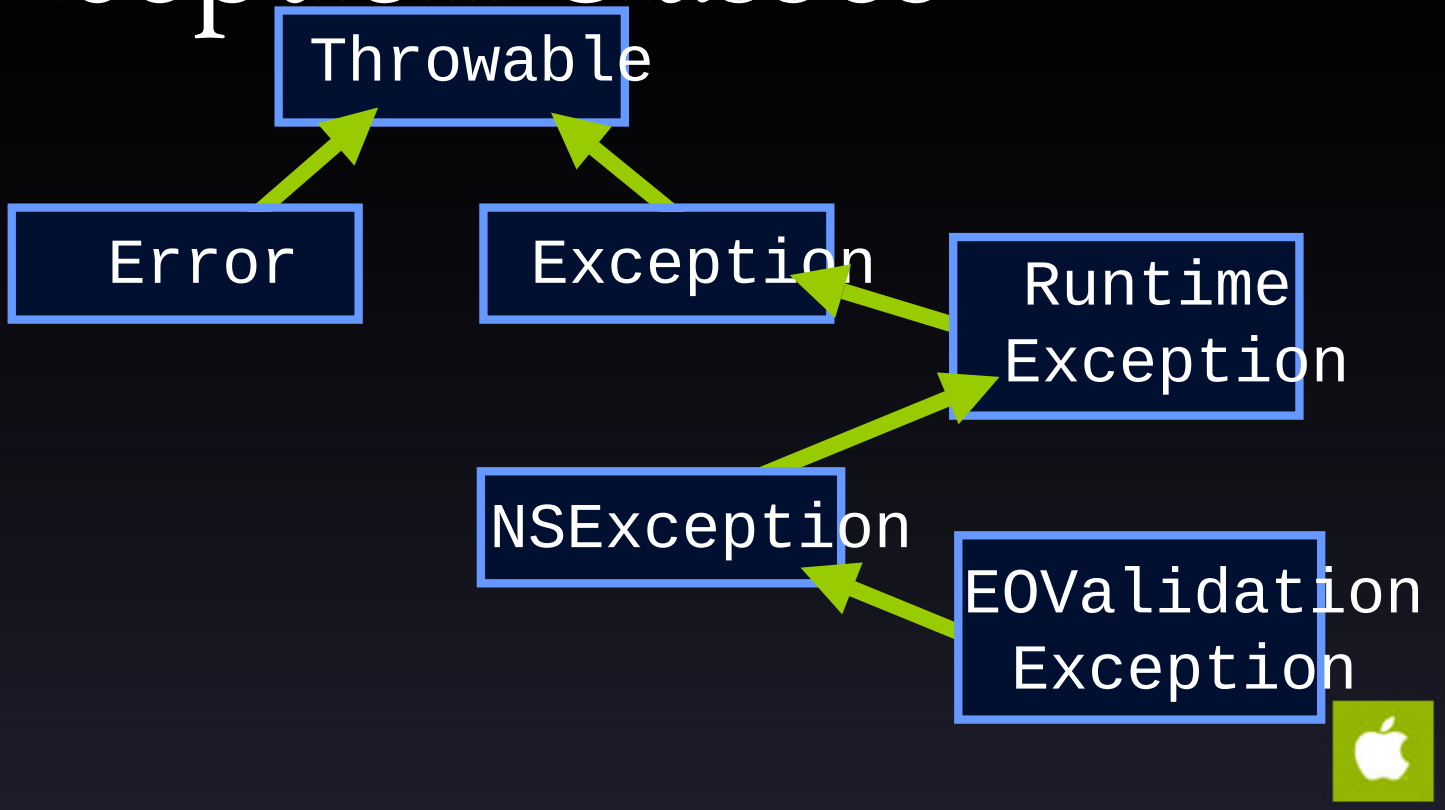
Finding Leaks

- Use ObjectAlloc
- Use MallocDebug
  (only on Mac OS X server)

Demo

Exceptions and Error Messages: How to Read Exceptions

Java and Apple Exception Classes



More on Exceptions

- Use gdb to break on [NSException raise]
- Study the stack trace leading to the raise
- Invoke getStackTrace() on NSException to print Java stack trace

Freed Objects Exception

- Use NSZombieEnabled (more in NSDebug.h)
- Set the flag inside your app, or inside gdb, or as an environment variable

Debugging WO Java Applications

- Incorrect CLASSPATH can make app fail
- Use OTHER_CLASSPATH and NSJavaUserPath to properly locate class files
- Look at default settings in JavaConfig.plist in $NEXT_ROOT/Library/Java

Debugging WO Java Applications (Cont.)

- Java StackOverflowError or OutOfMemoryError:
  - Modify the Java VM settings
- Potential memory leaks:
  - Avoid static or global references
- Beware of Java inner classes

Debugging WO Java Applications (Cont.)

- Use OptimizeIt to profile trouble spots
- Can download a demo version at http://www.optimizeit.com

Other Resources

- WO Support home page
  - http://www.apple.com/support/webobjects
  - Provide links to various helpful TIL articles and Tech Exchange
- Public mailing lists
  (eof, wof from the Omnigroup)
- AES Support
- How to report a bug with WebRadar

WebObjects Sessions

Fine-Tuning the Performance of WebObjects	Hall B **Wed., 4:00pm**
Complex WebObjects Application Design	Hall B **Thur., 9:00am**
WebObjects: Tips from the Experts	Hall B **Thur., 10:15am**
Advanced HTML/DHTML/ JavaScript Tips and Tricks	Hall B **Thur., 4:00pm**

Don’t Miss

Using Project Builder for Mac OS X	Hall A1 **Thur., 2:30pm**
Apple Development Tools for Mac OS X	Hall A1 **Thur., 4:00pm**

Debugging Contest

Q&A

Think different.™


99
Worldwide
Developers
Conference

BONUS SLIDES

- Parental Advice

Parental Advice* (1)

- Never assume something is too simple to be wrong—don't start with complex explanations
- When a hunch and a fact collide, the fact wins

* Thanks to Julie Zelenski for this Parental Advice from her presentation, "The Zen of Debugging"

Parental Advice (2)

- Be systematic
- Be persistent
- Don’t panic!
- Do not change your code haphazardly trying to track down a bug
- Look for one bug at a time

Parental Advice (3)

- If your code was working a minute ago, but now it doesn't—what was the last thing you changed?
- Be critical of your beliefs about your own code—it's usually impossible to see a bug in a method when your gut says that method is innocent

Parental Advice (4)

- Debugging depends on an objective and reasoned approach; late nights and long hours erode this ability—you'll debug much better on some sleep
- And, you'll code much better with sleep

Sample gdb Initialization File

```
set signal-exception 4
    handle SIGWINCH nostop ignore noprint
   define reason
   po [*((id *)($fp + 8)) reason]
   end
   document reason
   Print the reason for an NSException.
   When a program is inside of the method -[NSException raise], this
   command prints out a textual reason for the exception.
   end
   define self
   p/x *((id *)($fp + 8))
   end
```

Sample gdb Initialization File (Cont.)

```
document self
   Determines self for the current stack frame.
   end
   define po-self
   po *((id *)($fp + 8))
   end
   document po-self
   Prints out (sends -description to) self.
   end
   define self-class
   po NSStringFromClass((*((id *)($fp + 8))->isa))
   end
   document self-class
   Displays the name of self's class.
   end
```

Sample gdb Initialization File (Cont.)

```
document arg
Displays an argument for the current stack frame.
Takes a single parameter.  Argument zero is the selector
(which isn't very interesting).  Argument one is the first
actual argument.
end
define po-arg
po *((id *)($fp + 4 * ($arg0 + 3)))
end
document po-arg
Sends printForDebugger to an argument of the stack frame.
Like arg, the zero argument is the selector (don't do this).
Argument one is the first actual argument.
end
define super-class
po NSStringFromClass((*((id *)($fp + 8))->isa)->super_class)
end
document super-class
Displays the name of self's super class.
end
```

Welcome

To Advance through Presentation
Use Page Up and Page Down Keys