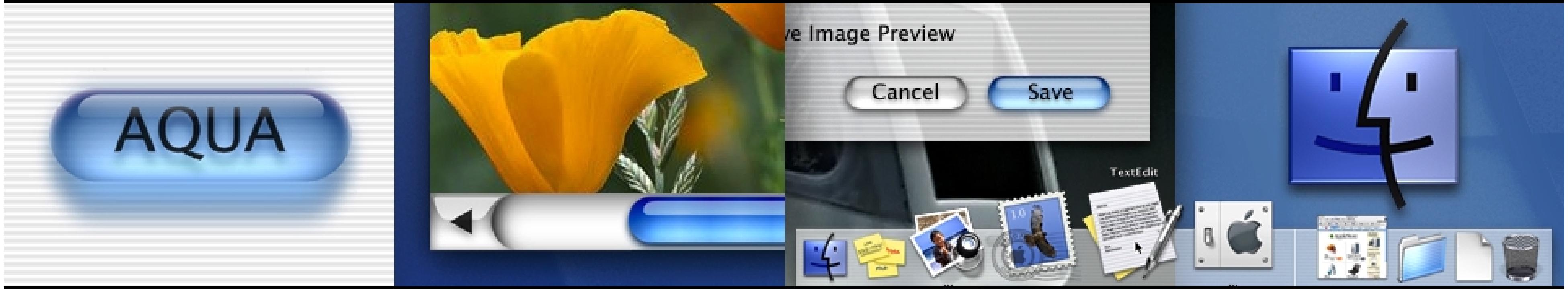
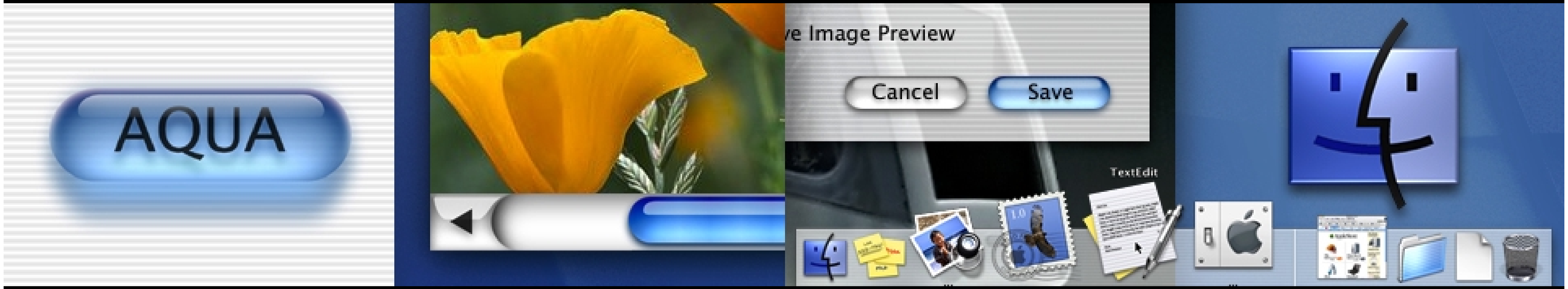
Session 416

Designing Reusable Components



Session 416

Designing Reusable Components



Kelly Kazem
Sr. Systems Engineer, Apple iServices

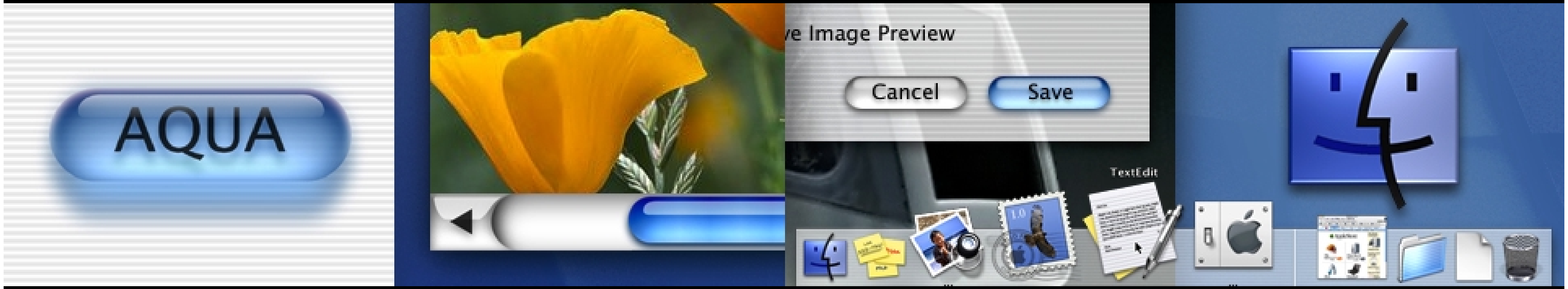
Introduction

- This Session:
  - Assumes some knowledge of WebObjects
  - Cover strategies for designing, debugging and optimizing reusable components
  - Uses Java-based examples
  - Has something for everyone!

What You’ll Learn

- Architecture Overview
- Packaging into Frameworks and Palettes
- Controlling Component Synchronization
- Using Shared State
- Static Components
- Dynamic Reusable Components
- Performance Tips
- Debugging Tips
- Pitfalls to Avoid
- Design Issues

Reusable Component Architecture
Packaging Reusable Components



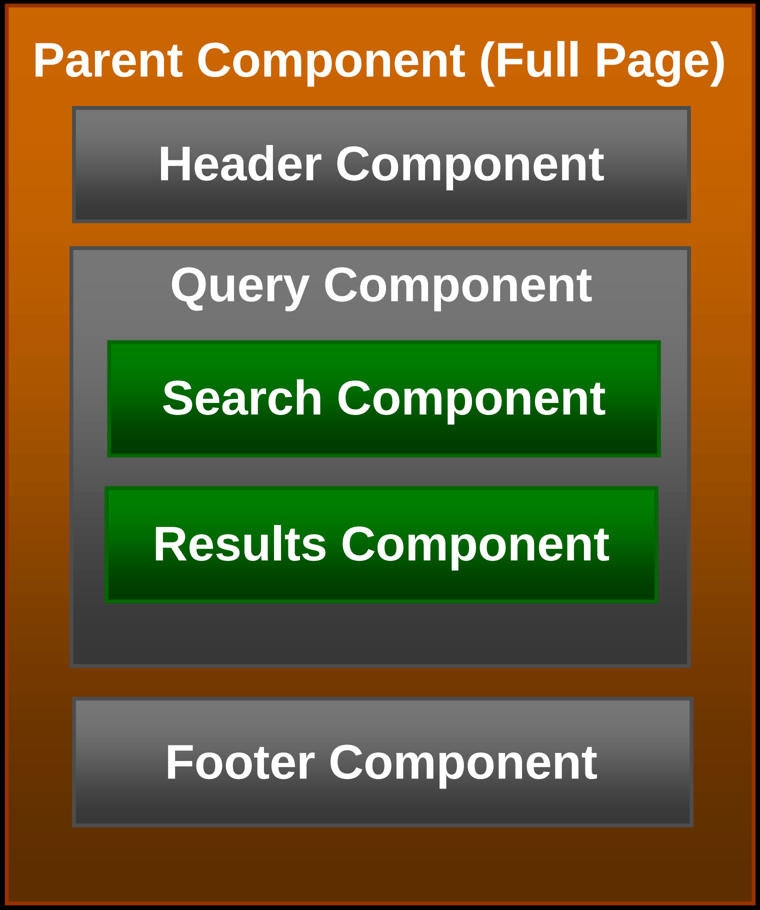
Scott Sweet
Systems Engineer, Apple iServices

Benefits of Reusable Components

- Organize application resources and logic into frameworks
- Encapsulate complex behavior without adding additional complexity to your application design
- Rapid application development through “assembly” of components
- Demanding requirements for personalization and interactive web apps
- Reuse

WebObjects Component Architecture

- WOComponent's can represent full pages or partial pages
- Reusable Components are nested in a parent or sub-component
- WebObjects Builder has excellent support for creating Reusable Components



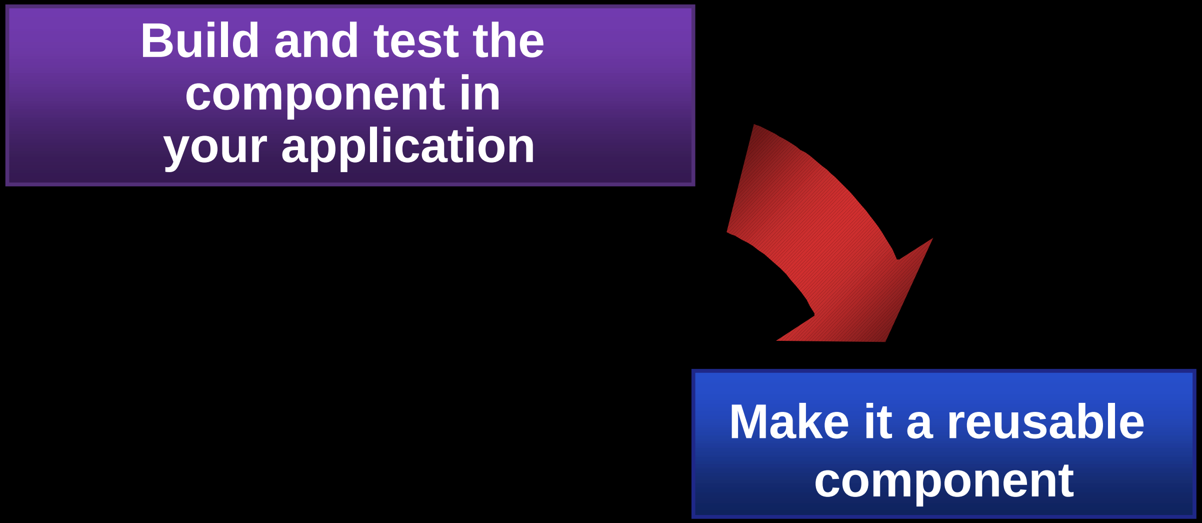
Building Reusable Components

The Process

Build and test the component in your application

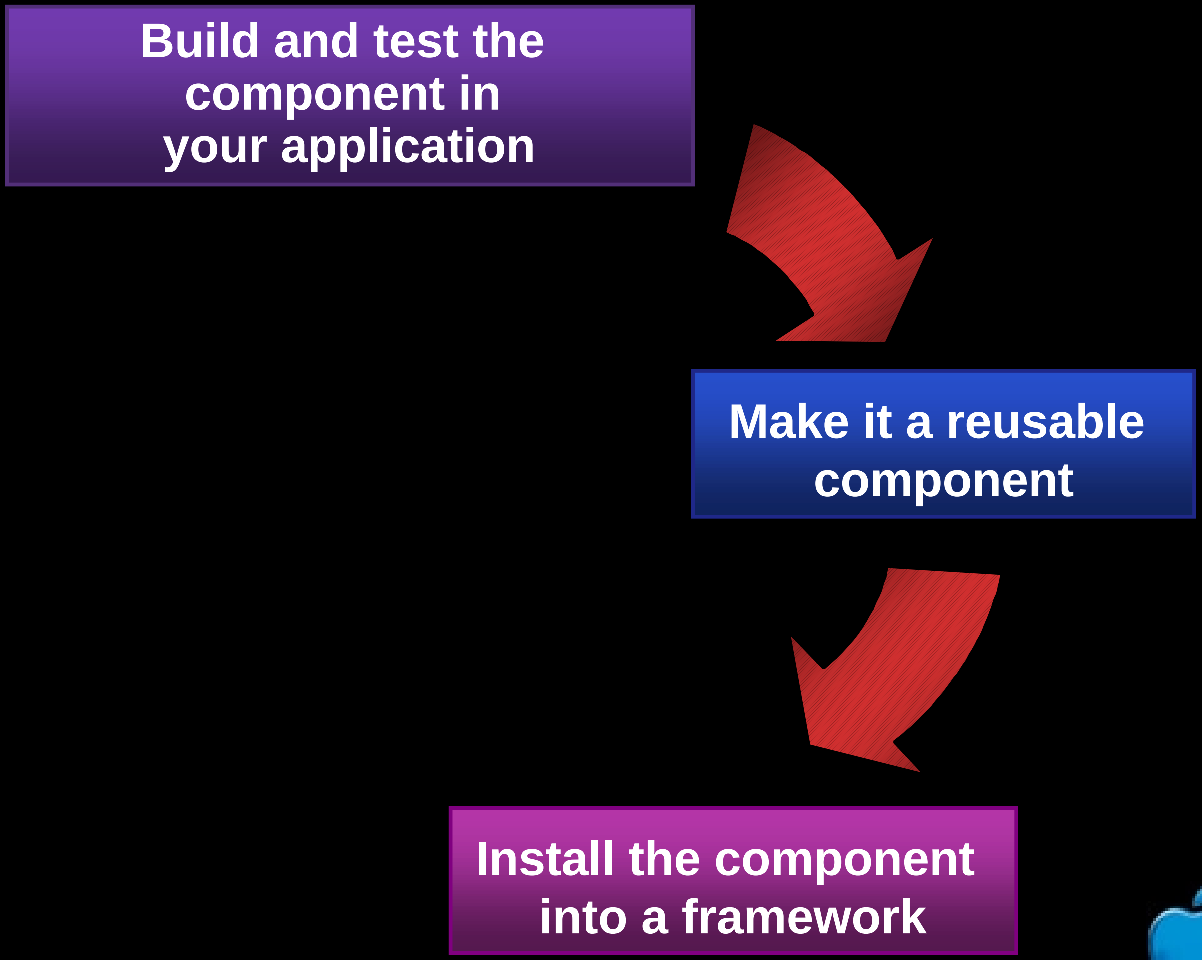
Building Reusable Components

The Process


Build and test the component in your application
Make it a reusable component

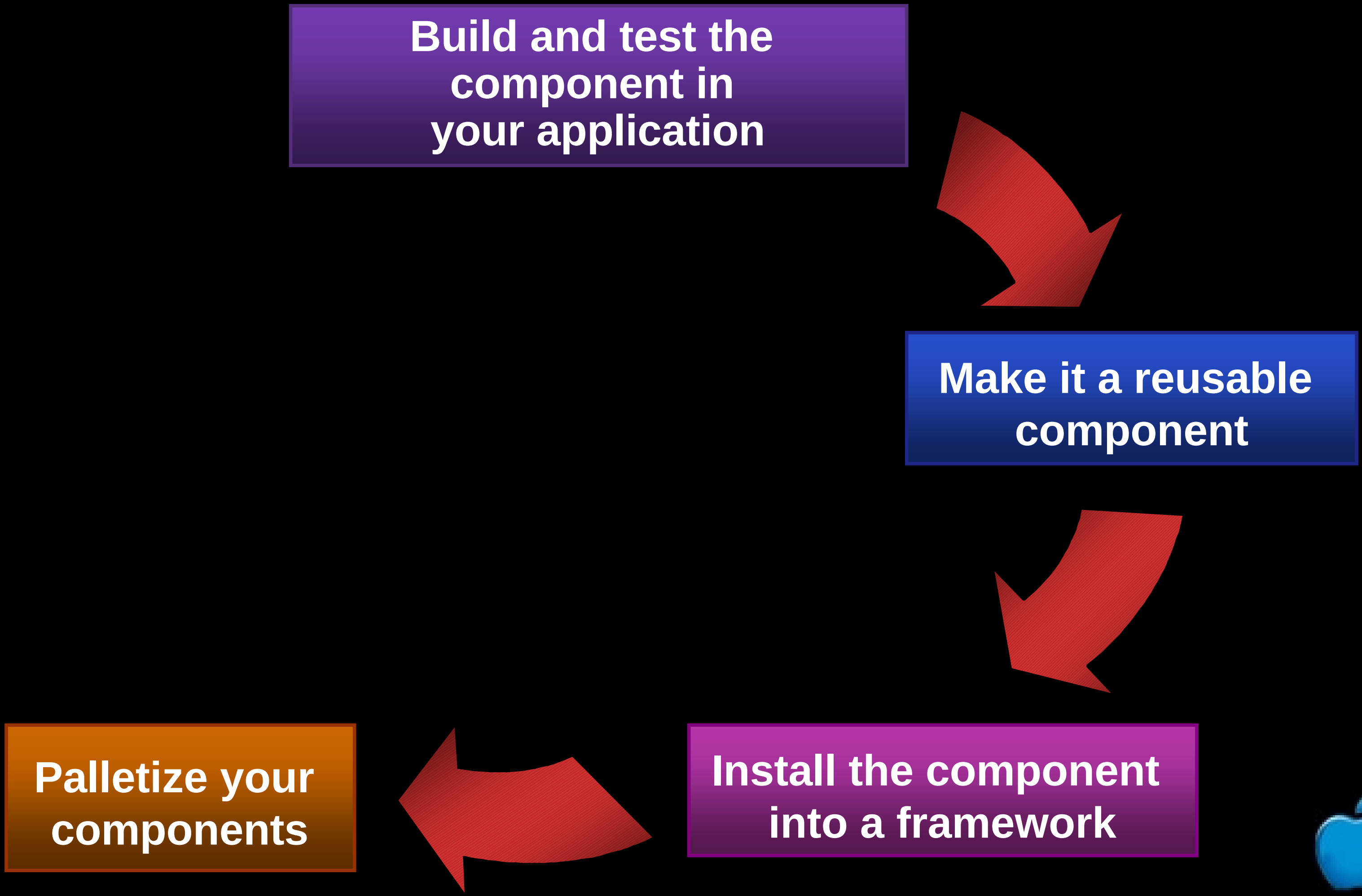
Building Reusable Components

The Process


Build and test the component in your application
Make it a reusable component
Install the component into a framework

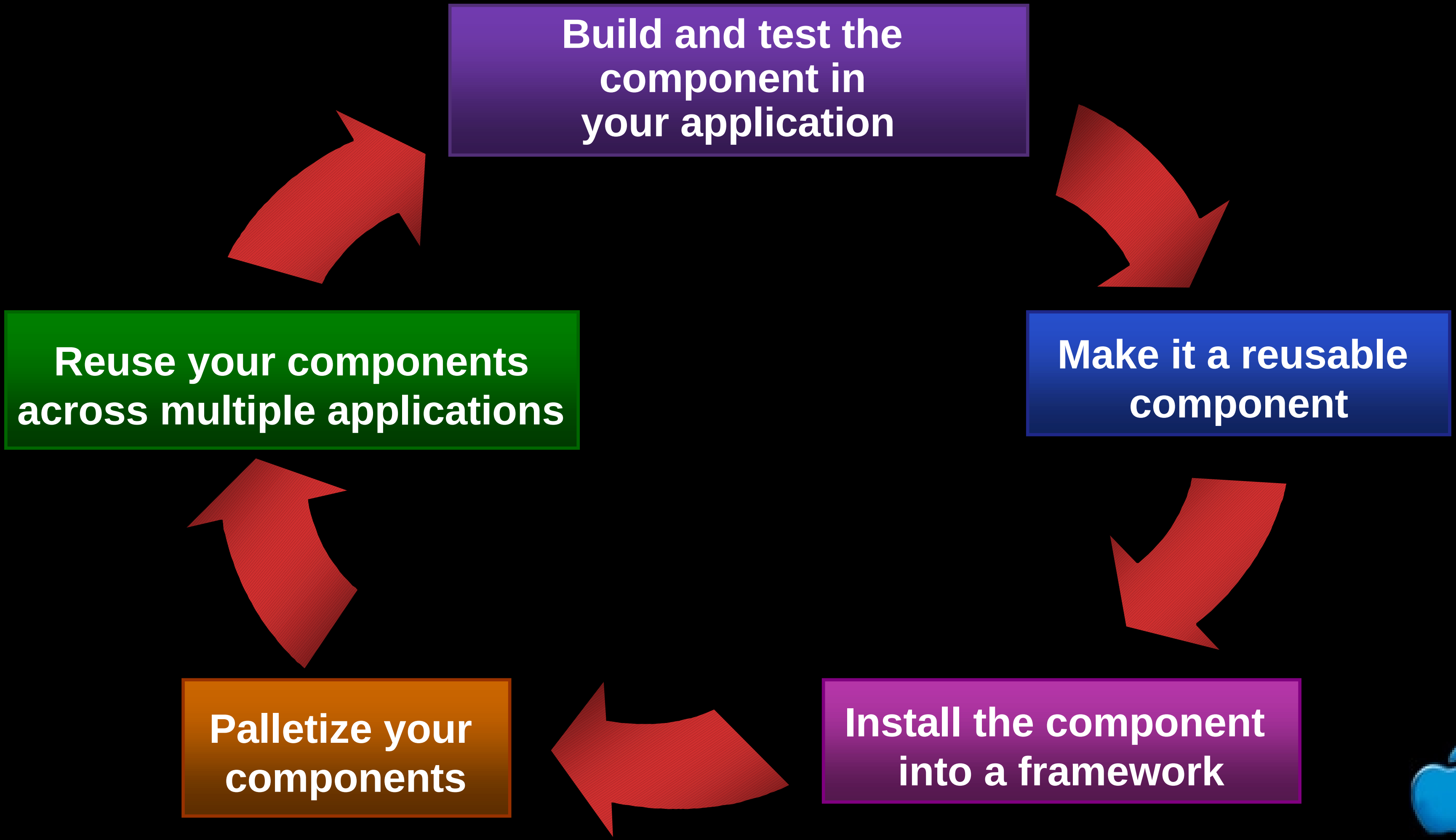
Building Reusable Components

The Process


Build and test the component in your application
Make it a reusable component
Install the component into a framework
Palletize your components

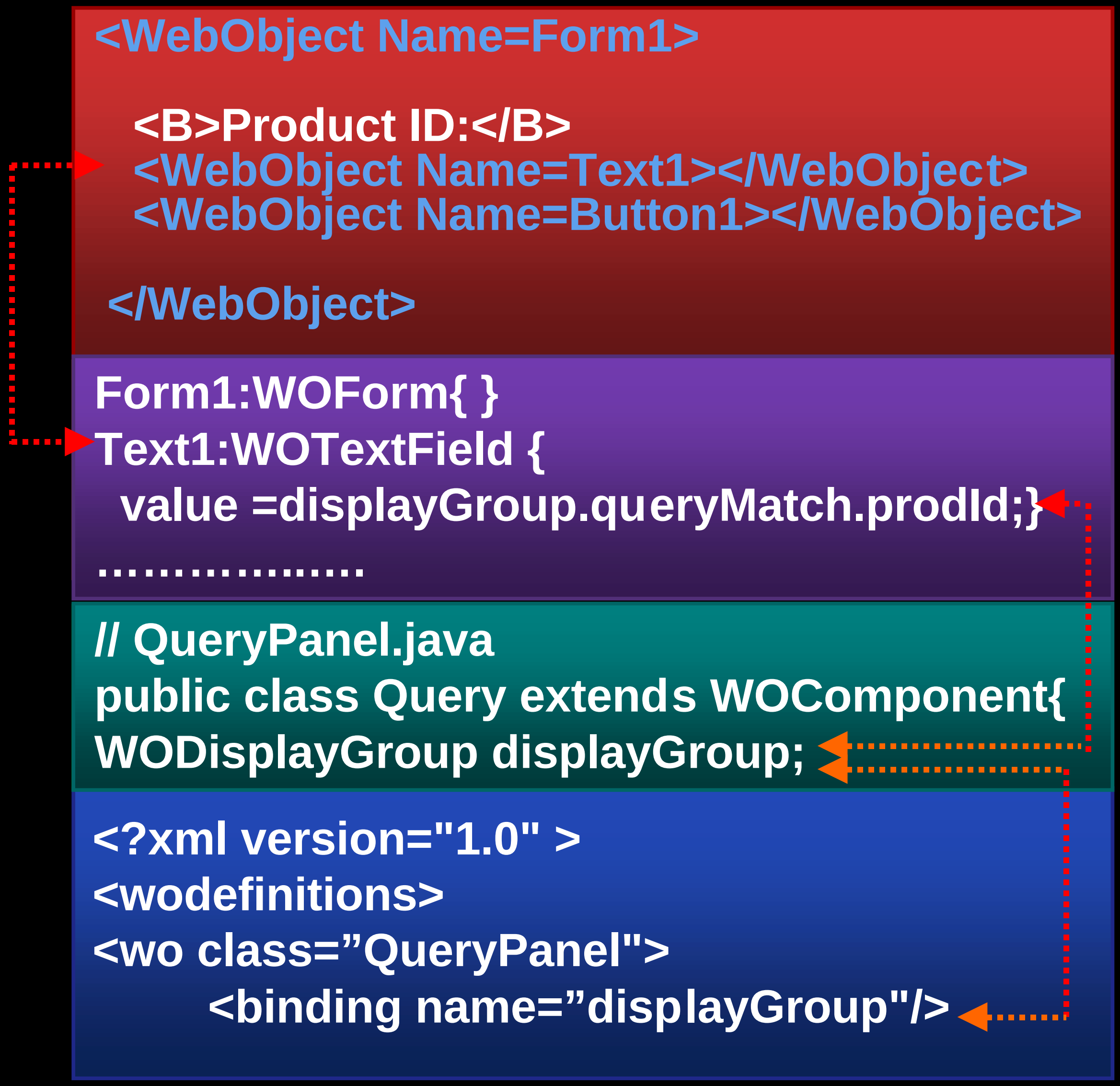
Building Reusable Components

The Process



Build and test the component in your application
Make it a reusable component
Install the component into a framework
Palletize your components
Reuse your components across multiple applications

Reusable Component



```
<WebObject Name=Form1>

   <B>Product ID:</B>
   <WebObject Name=Text1></WebObject>
   <WebObject Name=Button1></WebObject>

 </WebObject>
```

HTML Template(.html)
(partial HTML document)

```
Form1:WOForm{ }
Text1:WOTextField {
  value =displayGroup.queryMatch.prodId;}
………………
```

Bindings File(.wod)

```
// QueryPanel.java
public class Query extends WOComponent{
WODisplayGroup displayGroup;
```

Component Class

```
<?xml version="1.0" >
<wodefinitions>
<wo class=”QueryPanel">
        <binding name=”displayGroup"/>
```

API File(.api)

Intercomponent Communication

- Parent and child components typically communicate via synchronization of state variables
- State variables are automatically synchronized before and after each phase of the request-response loop

```
//in parent component
public class Main extends ….
String lastName;
…………..
……
```

```
//in child component
public class NameLookup extends ...
String aName;
…………..
……
```

```
//parents .wod file
Comp1:NameLookup{  aName = lastName ; }
……………
```

Synchronization

Parent Callbacks From Child

- Child components invoke actions in the Parent using a callback method name property and performParentAction method

```
//in parent component
public WOComponent recalc(){
…………………
}
```

```
//in parents .wod file
Child1:MyComponent{ calculateAction = “recalc”; }
```

```
// in the child component
String calculateAction;  //the method name property
WOComponent recalculateTotal()
{       return performParentAction(calculateAction);   }
```

Parent-Child Communication

- Child also has access to the Parent through the parent() method

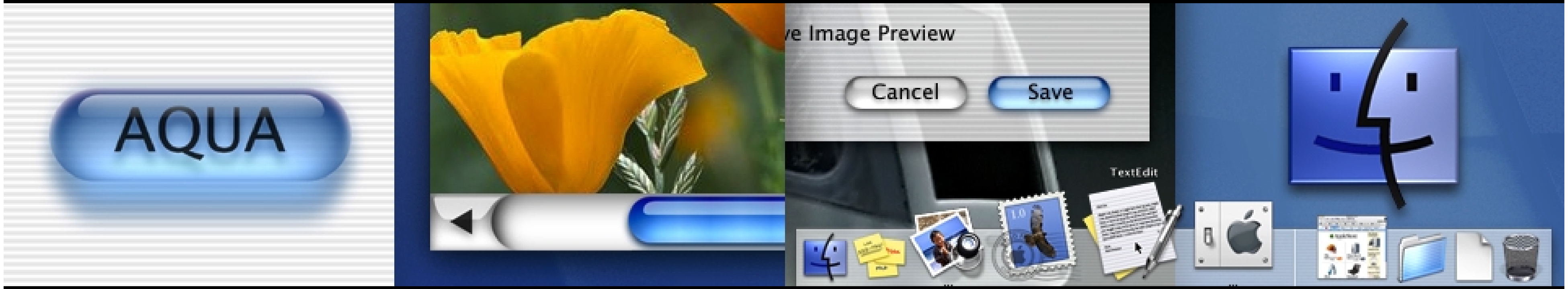
Example 1.

```
// in the child component
String  getParentName()
{
       return parent().name();
}
```

Example 2.

```
// in the child component
String  getAccountName()
{
      return parent().valueForKey(“accountName”);
}
```

Demo



Reusable Components

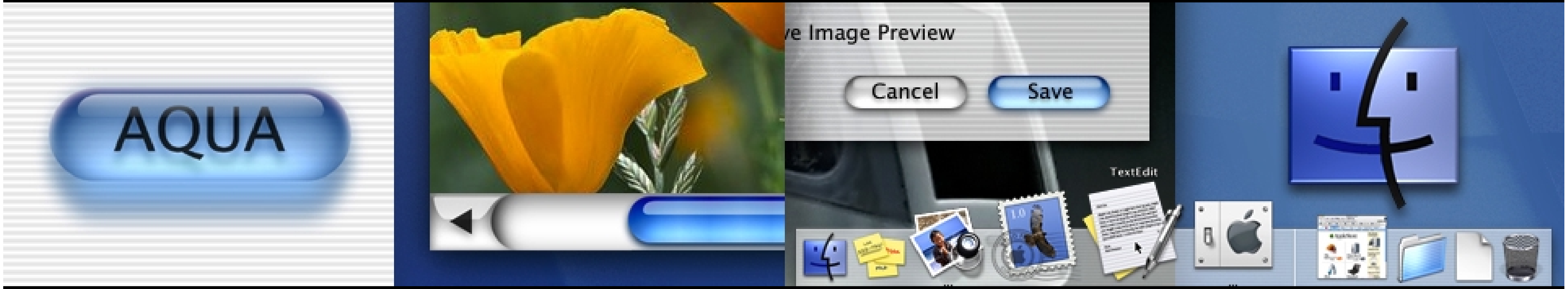
Packaging Reusable Components

- Components can be packaged into frameworks
  - Debug and optimize the components in a WebObjects Application project first!
  - Use the “WebObjects Framework” project type
  - Framework contains all your components resources
  - “Make install” to compile and install the framework
  - Optionally “palletize” the components

Palletizing Reusable Components

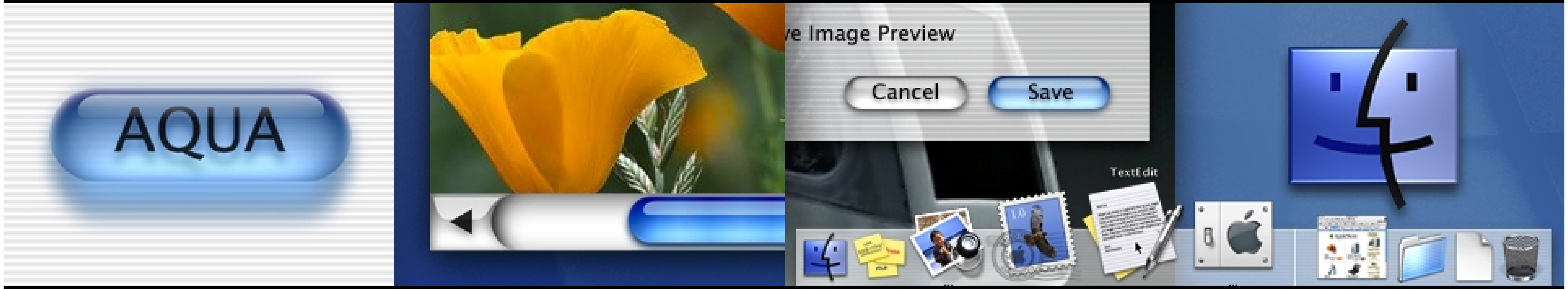
- Benefits
  - Developers can create several custom palettes
  - Palettes can be shared by developers
- The WOB Palette helps organize and simplify access to custom reusable components
  - Simply drag components from PB onto the palette
  - Drag a custom image over the default image on the palette
  - You can specify a custom image to render your component in WOB “design view”

Demo



Making a Reusable Framework and Palletizing Components

Intercomponent Synchronization



Kelly Kazem
Sr. Systems Engineer, Apple iServices

Intercomponent Synchronization

- Two Types
  - Automatic(default)
    - No coding to implement
    - Possible performance implications
    - Possible unwanted side effects
  - Manual
    - Some coding and thought needed
    - More control

Intercomponent Synchronization

- Attributes synchronized up to 6 times per request-response cycle
  - That’s potentially 12 method invocations per binding per component per request!
  - Easy to introduce unwanted side effects in accessors

```
for( each Element in graph )
{
    synchronize bindings
    takeValuesFromRequest
    synchronize bindings
}

for( each Element in graph )
{
    synchronize bindings
    invokeAction
    synchronize bindings
}

for( each Element in graph )
{
    synchronize bindings
    appendToResponse
    synchronize bindings
}
```

Intercomponent Synchronization (1)

This example has 3 problems. Can you find them?

```
//in parent component
String userName;

……….
………….
……………….
```

```
//parents .wod file
Comp1:BeautifyString{
     aString = userName; }
```

```
//in child component
String aString;

public String getAString(){
String s=“”;


   s = aString.substring(0,1).toUpperCase();
   s = s + aString.substring(1).toLowerCase();

  return s;
}
```

```
//childs .wod file
String1:WOString{
            value = aString; }
```

Intercomponent Synchronization (2)

First fix: state variable initialization check

```
//in parent component
String userName;

……….
………….
……………….
```

```
//in child component
String aString;

public String getAString(){
String s=“”;

 if( aString ! = null){
   s = aString.substring(0,1).toUpperCase();
   s = s + aString.substring(1).toLowerCase();
  }
  return s;
}
```

```
//parents .wod file
Comp1:BeautifyString{
     aString = userName; }
```

```
//childs .wod file
String1:WOString{
             value = aString; }
```

Intercomponent Synchronization (3)

Second fix: Disable synchroniztion

```
//in parent component
String userName;

……….
………….
……………….
```

```
// in child component
String aString;

public boolean synchronizesVariablesWithBindings(){
        return false;
}
public String getAString(){
String s;

 aString = (String)valueForBinding("aString");
 if( aString ! = null){
   s = aString.substring(0,1).toUpperCase();
   s = s.substring(1).toLowerCase();
  }
  return s;
}
```

```
//parents .wod file
Comp1:BeautifyString{
     aString = userName; }
```

```
//childs .wod file
String1:WOString{
          value = aString;  }
```

Intercomponent Synchronization (4)

Eliminate state variable

```
//in parent component
String userName;

……….
………….
……………….
```

```
// in child component
~~String aString;~~

public boolean synchronizesVariablesWithBindings(){
        return false;
}
public String getAString(){
String s;

 aString = (String)valueForBinding(“aString”);
 if( aString ! = null){
   s = aString.substring(0,1).toUpperCase();
   s = s.substring(1).toLowerCase();
  }
  return s;
}
```

```
//parents .wod file
Comp1:BeautifyString{
     aString = userName; }
```

```
//childs .wod file
String1:WOString{
           value = aString;  }
```

Intercomponent Synchronization (5)

Pushing value back to parent attribute

```
//in parent component
String userName;

……….
………….
……………….
```

```
// in child component

public boolean synchronizesVariablesWithBindings(){
                return false;
}
public String getAString(){
String s;

 s = (String)valueForBinding("aString");
 if( s ! = null){
   s = s.toLowerCase();
   s = s.substring(0,1).toUpperCase()+s.substring(1);
   this.setValueForBinding( s, "aString");
  }
  return s;
}
```

```
//parents .wod file
Comp1:BeautifyString{
     aString = userName; }
```

```
//childs .wod file
String1:WOString{
              value = aString;  }
```

Intercomponent Synchronization (6)

Cache attribute in state variable

```
//in parent component
EOEnterpriseObject cust;

……….
………….
……………….
```

```
// in child component


public boolean synchronizesVariablesWithBindings(){
            return false;
}
public EOEnterpriseObject getCustObject(){

  return valueForBinding( “custObject” );


  }
```

```
//parents .wod file
Comp1:CustDisplay{
     custObject = cust; }
```

```
//childs .wod file
String1:WOString{  value = custObject.name;  }
String2:WOString{  value = custObject.address;  }
……………….
```

Intercomponent Synchronization (6)

Cache attribute in state variable

```
//in parent component
EOEnterpriseObject cust;

……….
………….
……………….
```

```
// in child component
EOEnterpriseObject custObject; //using as cache


public boolean synchronizesVariablesWithBindings(){
            return false;
}

public EOEnterpriseObject getCustObject(){
   if( custObject == null )
  return valueForBinding( “custObject” );
      custObject =valueForBinding( “custObject” );
   return custObject;
  }}


// reset custObject to null  after appendToResponse
```

```
//parents .wod file
Comp1:CustDisplay{
     custObject = cust; }
```

```
//childs .wod file
 ……………………
String1:WOString{  value = custObject.name;  }
String2:WOString{  value = custObject.address;  }
……………….
```

Intercomponent Synchronization (7)

‘Caret Notation’

```
//in parent component
WODisplayGroup dg;

……….
………….
……………….
```

```
// in child component

public boolean synchronizesVariablesWithBindings(){
        return false;
}


public WODisplayGroup displayGroup(){

    return valueForBinding( “displayGroup” );
}
```

```
//parents .wod file
Comp1:ResultsCount{
    displayGroup = dg; }
```

```
//childs .wod file
String1:WOString{
    value =  displayGroup.allObjects.count;  }
```

Intercomponent Synchronization (7)

‘Caret Notation’

```
//in parent component
WODisplayGroup dg;

……….
………….
……………….
```

```
// in child component

public boolean synchronizesVariablesWithBindings(){
        return false;
}
```

~~public WODisplayGroup displayGroup(){~~

~~return valueForBinding(“displayGroup”);~~
~~}~~

```
//parents .wod file
Comp1:ResultsCount{
    displayGroup = dg; }
```

```
//childs .wod file
String1:WOString{
    value = ^displayGroup.allObjects.count; }
```

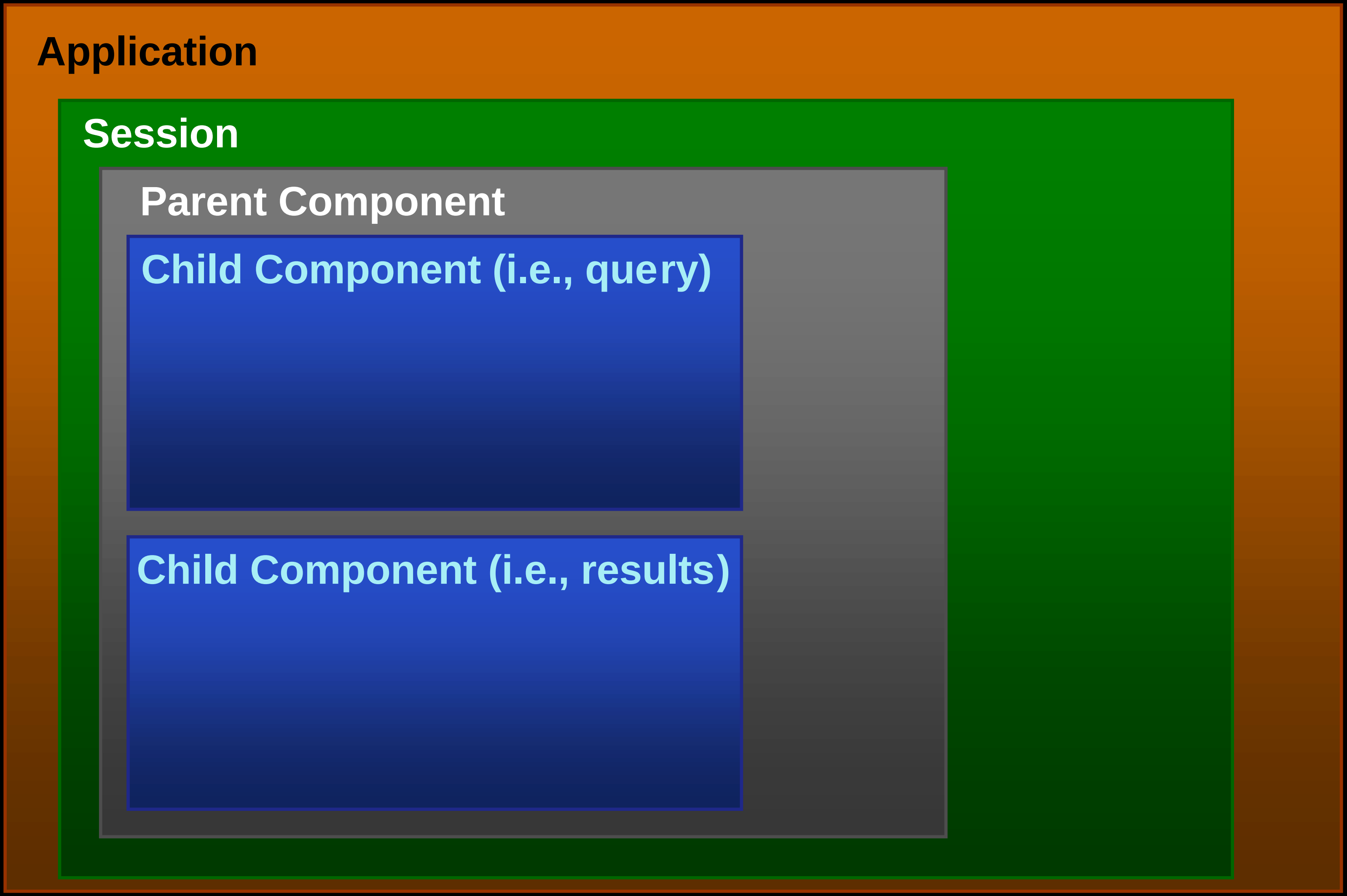
‘Stateless’ Components

- More efficient, component instance(s) are shared
- Smaller memory footprint
- No state beyond the current request-response loop
- Several included in WOExtensions

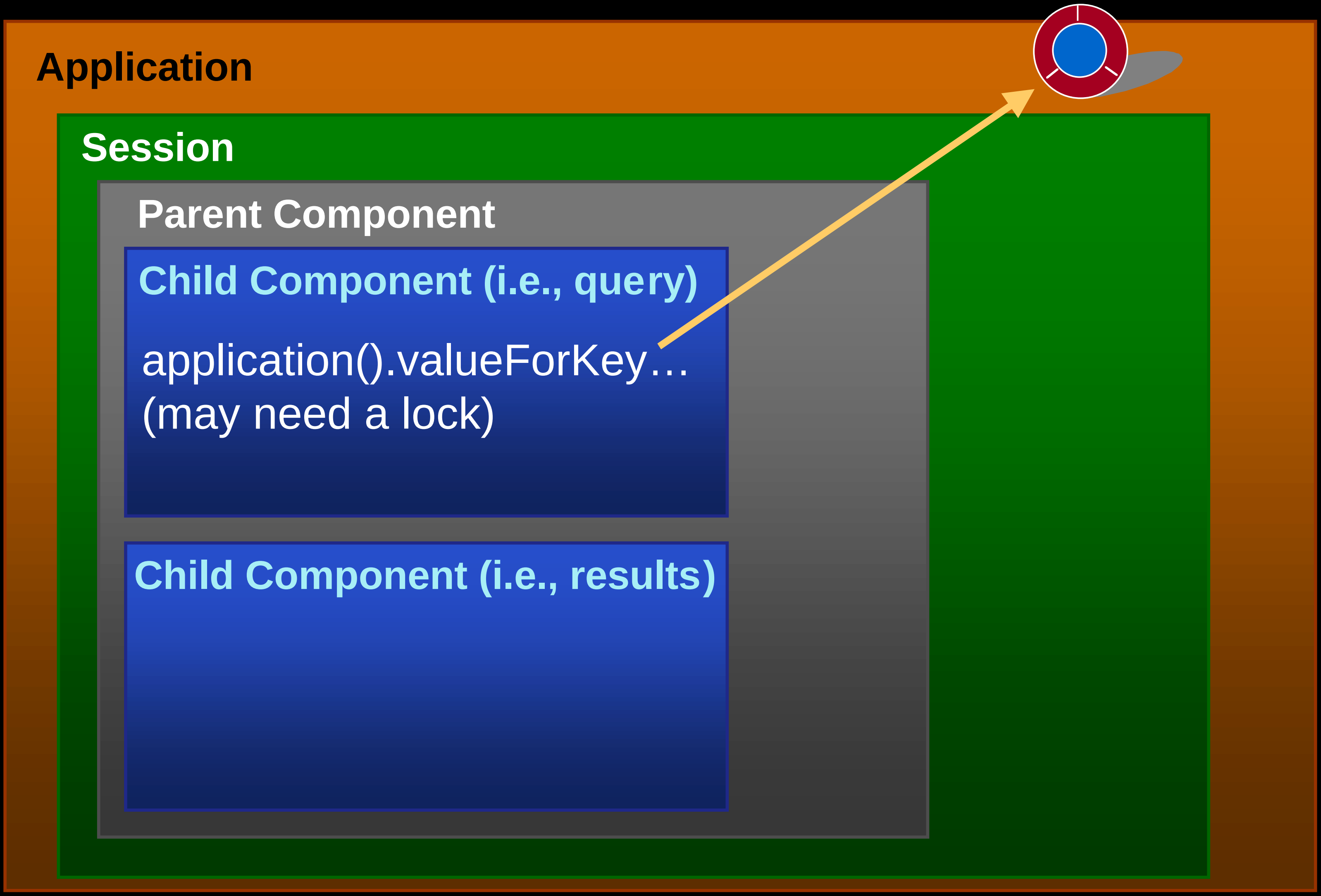
```
// in child component
public boolean isStateless() {    return true;     }

public void reset(){
            String someVar = null; // null out all instance variables
            …………..
}
```

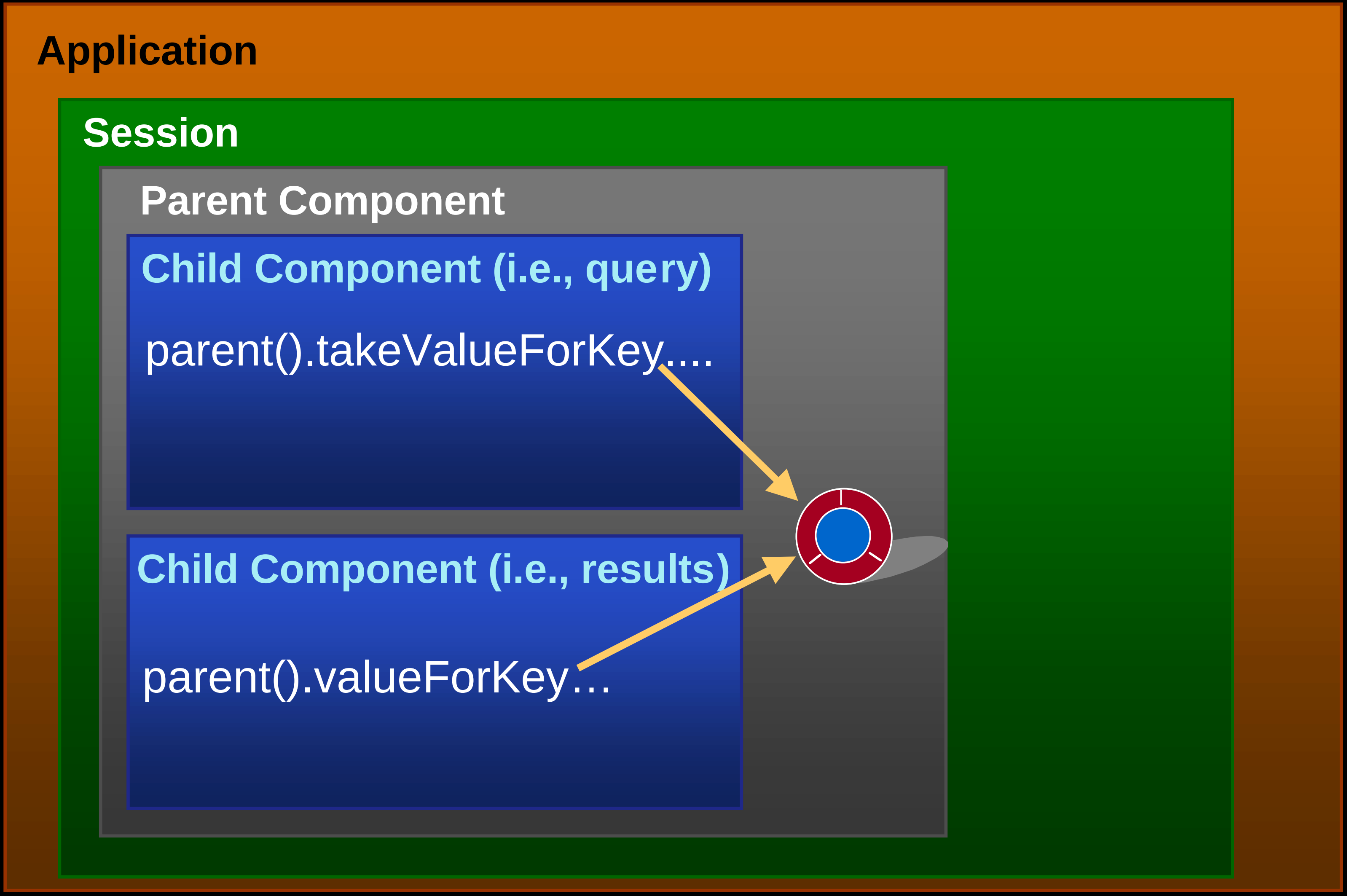
Shared State Strategies


Application
Session
Parent Component
Child Component (i.e., query)
Child Component (i.e., results)

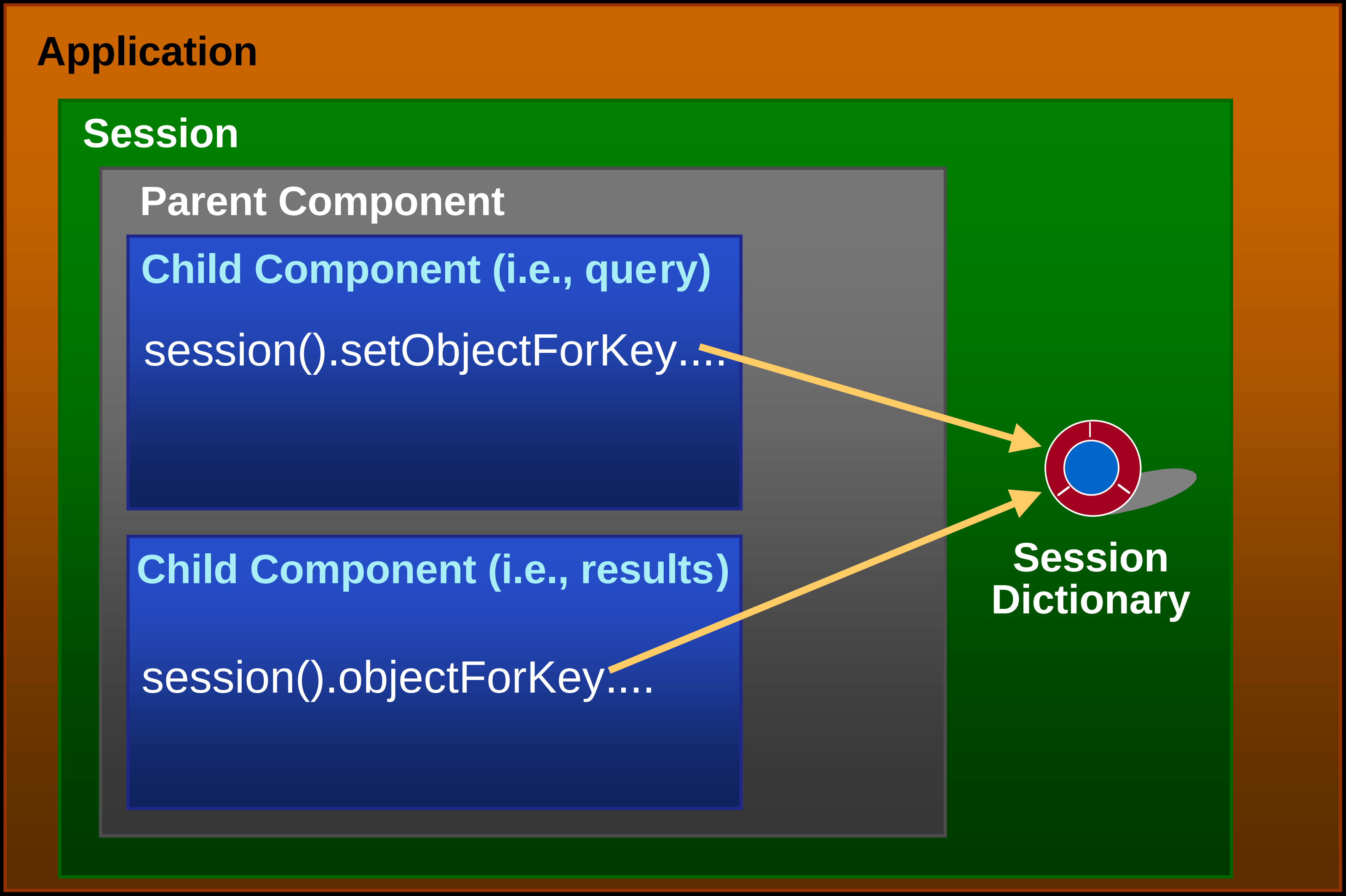
Shared State Strategies


Application
Session
Parent Component
Child Component (i.e., query)
application().valueForKey…
(may need a lock)
Child Component (i.e., results)

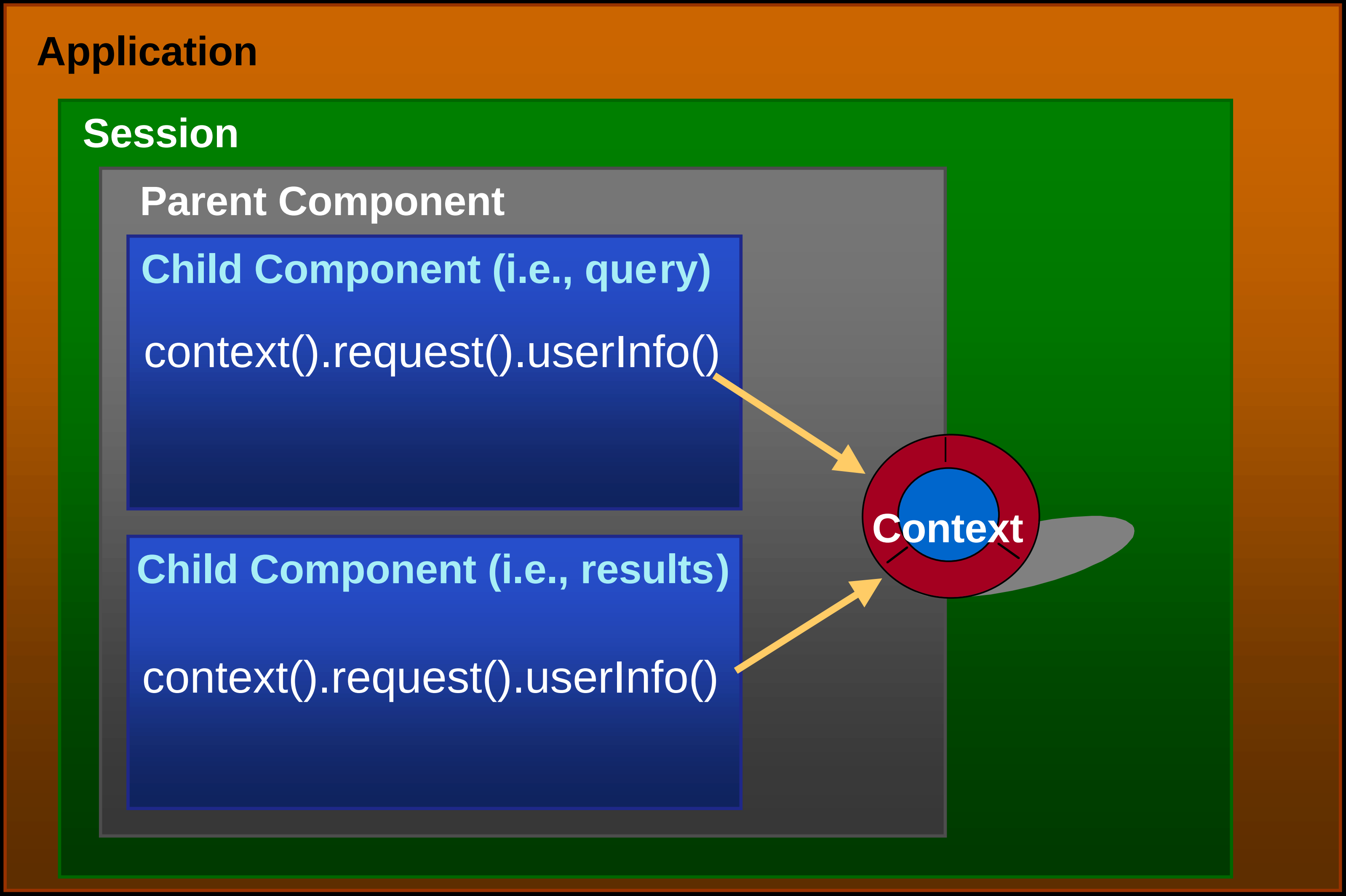
Shared State Strategies


Application
Session
Parent Component
Child Component (i.e., query)
parent().takeValueForKey....
Child Component (i.e., results)
parent().valueForKey…

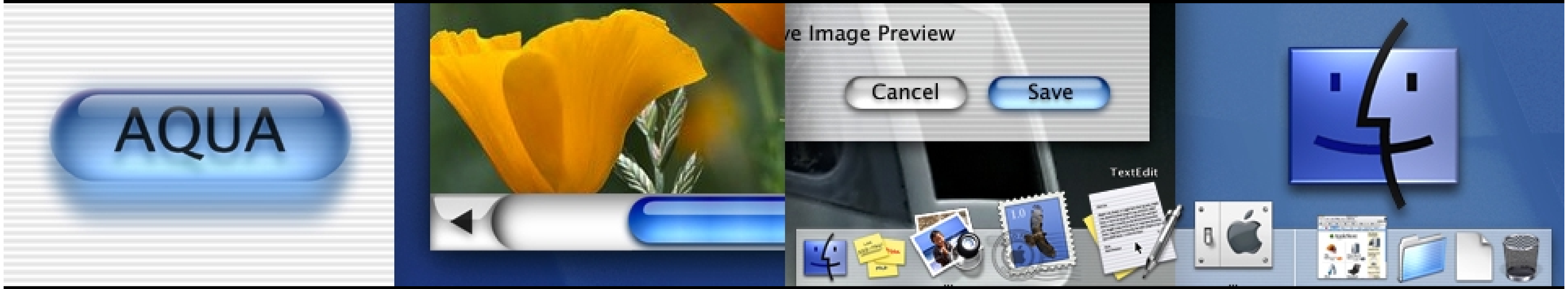
Shared State Strategies


Application
Session
Parent Component
Child Component (i.e., query)
session().setObjectForKey....
Child Component (i.e., results)
session().objectForKey....
Session Dictionary

Shared State Strategies


Application
Session
Parent Component
Child Component (i.e., query)
context().request().userInfo()
Child Component (i.e., results)
context().request().userInfo()
Context

Dynamic Reusable Components



Bob Frank
Consulting Engineer, Apple iServices

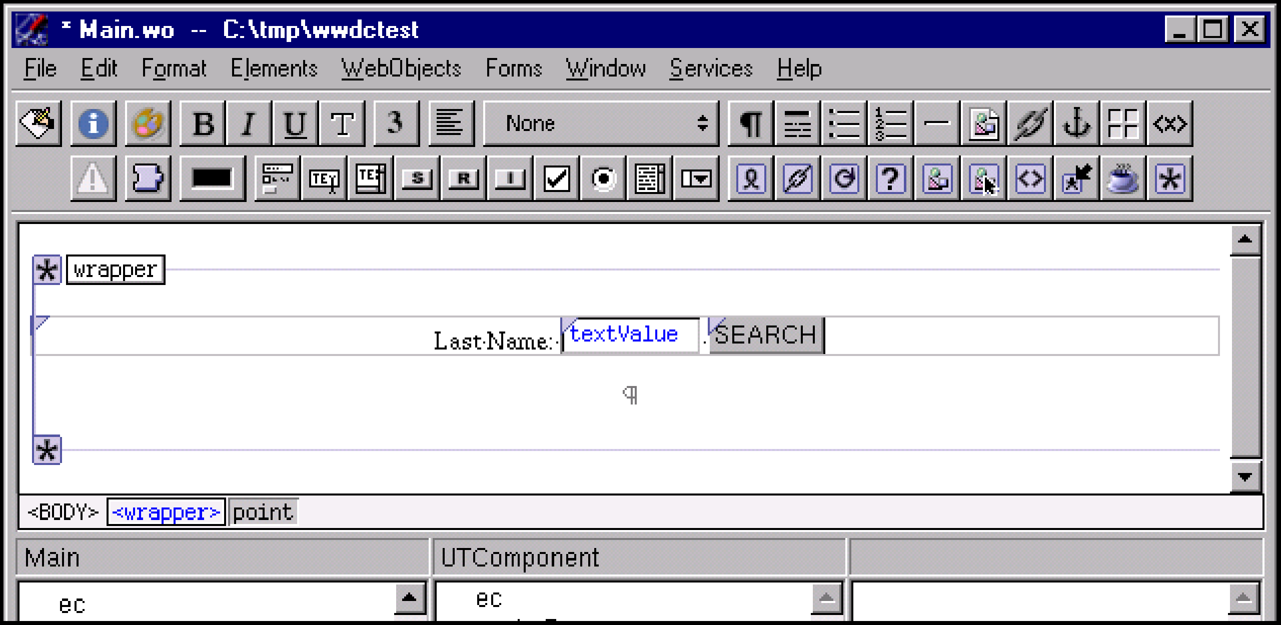
Dynamic Reusable Components

- WOSwitchComponent
  - Allows you to dynamically select which component to render at runtime

```
// in parent’s .wod file
SwitchComponent:WOSwitchComponent {
     WOComponentName = someComponentName;
          binding1= value;  ……….;
}
```

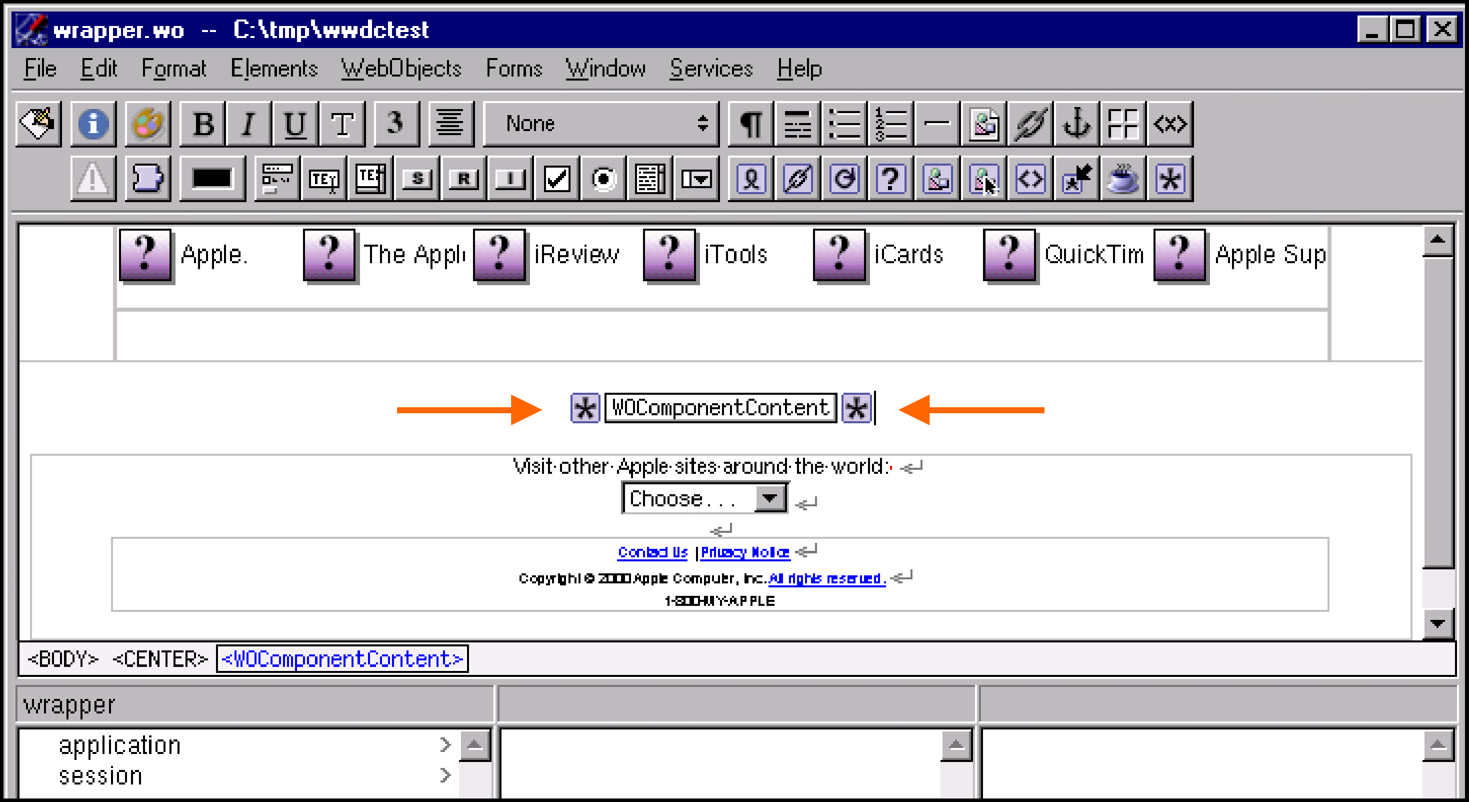
Using WOComponentContent

Your reusable is "wrapped" by your "Wrapper" component

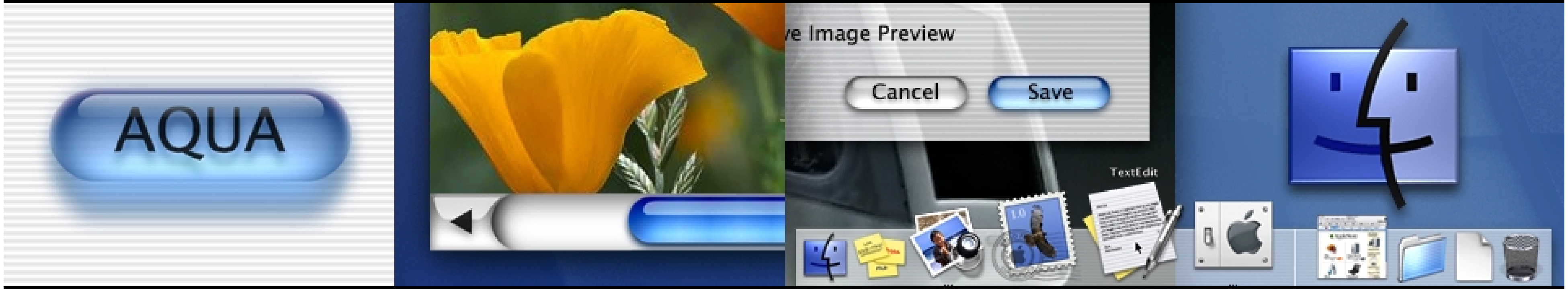

* Main.wo -- C:\tmp\wwdctest
File Edit Format Elements WebObjects Forms Window Services Help
None
wrapper
Last Name: textValue SEARCH
<BODY> <wrapper> point
Main
UTComponent
ec
ec

Dynamic Reusable Components

WOComponentContent is used to create your wrapper component


wrapper.wo -- C:\tmp\wwdctest
File Edit Format Elements WebObjects Forms Window Services Help
None
Apple. The Appl iReview iTools iCards QuickTim Apple Sup
WOComponentContent
Visit other Apple sites around the world:
Choose . . .
Contact Us | Privacy Notice
Copyright © 2000 Apple Computer, Inc. All rights reserved.
1-800-MY-APPLE
<BODY> <CENTER> <WOComponentContent>
wrapper
application
session

DEMO

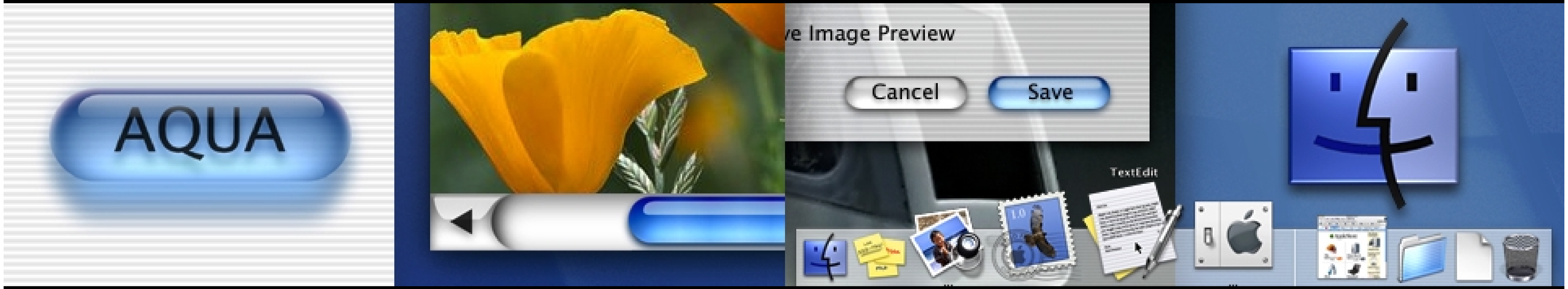


Dynamic Wrappers

Subclass WOComponent

- Create a personal tool-kit
- Example methods
  - Expire/NoCache specific page
  - goToPreviousPage
  - Store state (state() & setState())
  - nullAction()

Performance and Debugging

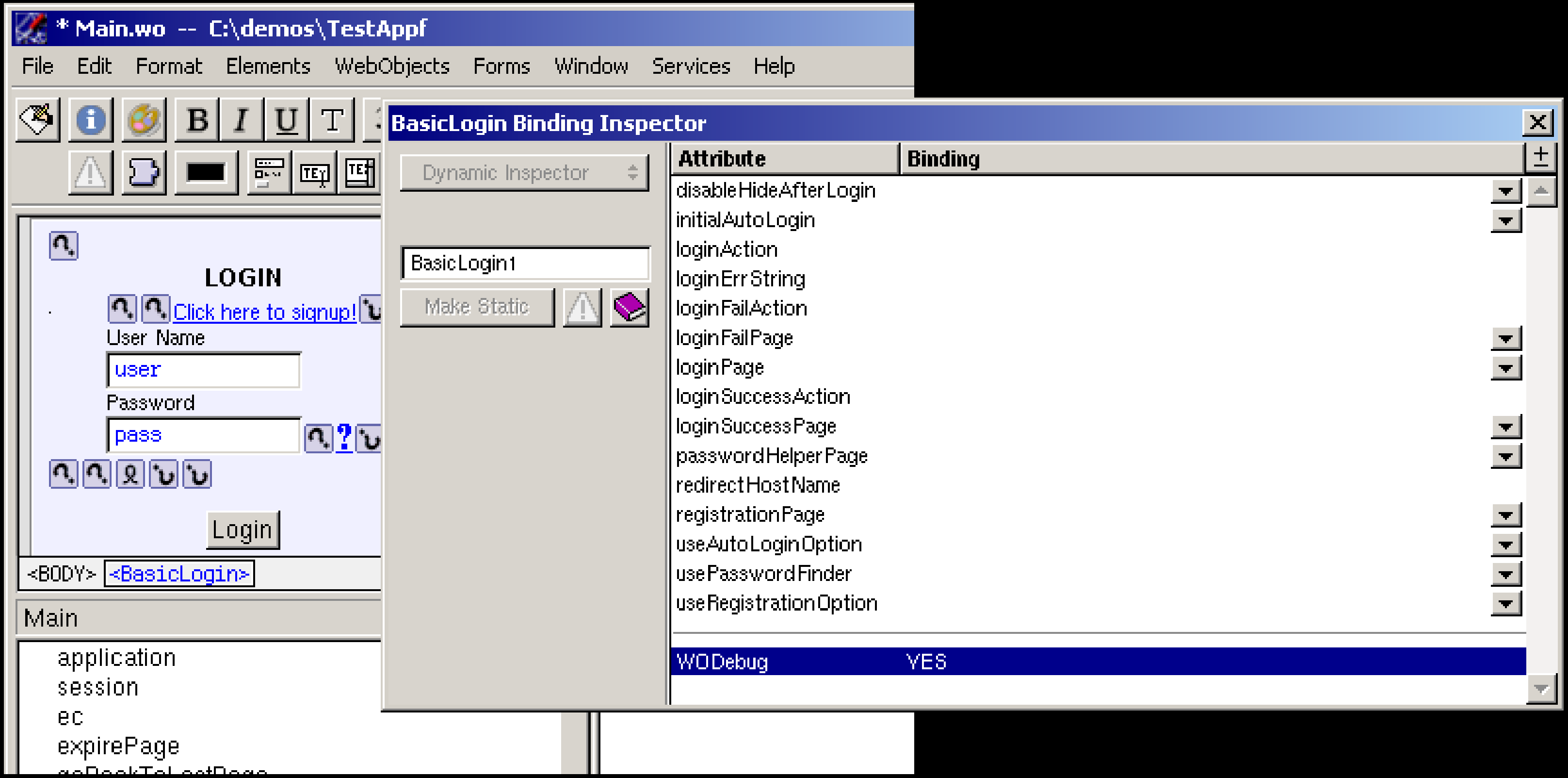


Kelly Kazem
Sr. Systems Engineer, Apple iServices

Debugging Tips

- GDB/JDB (make sure to set: OTHER_JAVATOOL_FLAGS = -g) and enable both debuggers for java!
- Good ol’ System.out.println( );
  - logString();
  - debugString(); with -WODebuggingEnabled YES
- Use comments to isolate HTML
- WODebug binding attribute

Setting WODebug



Attribute	Binding
disableHideAfterLogin	
initialAutoLogin	
loginAction	
loginErrString	
loginFailAction	
loginFailPage	
loginPage	
loginSuccessAction	
loginSuccessPage	
passwordHelperPage	
redirectHostName	
registrationPage	
useAutoLoginOption	
usePasswordFinder	
useRegistrationOption	
WODebug	YES

WODebug Console Output

```
[AnyField:WOAnyField] keyList <== (aKey2: (name, symbol))
[AnyField:WOAnyField] selectedKey <== (selectedCountry: *nil*)
[AnyField:WOAnyField] (key: "name") ==> aKey
[AnyField:WOAnyField] displayKey <== (NSInlineCString: "name")
[AnyField:WOAnyField] sourceEntityName <== (NSInlineCString: "Country")
[AnyField:WOAnyField] (key: "symbol") ==> aKey
[AnyField:WOAnyField] displayGroup <== (dg: <WODisplayGroup: 0x2753ca0>)
```

```
// bindings file
AnyField: WOAnyField {
                    keyList = aKey2;
                    selectedKey = selectedCountry;
                    key = aKey;
                    displayGroup = dg;
                    sourceEntityName = "Country";
                    displayKey = "name";
                    WODebug = YES;}
```

Performance Tips

- Disable binding synchronization
- Stateless components
- Enable component definition caching

```
setCachingEnabled( true ); //in application init code
```

or on the command line:

```
-WOCachingEnabled YES
```

- Disable various debugging modes
- Use WOEvent Logging new in 4.5!

http://myhost/cgi-bin/WebObjects/App.woa/wa/WOEventSetup

Pitfalls to Avoid

- takeValuesFromRequest isn’t always invoked
- Initializing variables too early
- Returning self/this from child components
- Side effects in accessor methods
- Nested <FORM> tags
  - “Conditional Form” example tarball at
  - **http://enterprise.apple.com/wwdc2000/416/ConditionalForm.gt**
- Child component not a partial document
- Retain cycles

Design Issues

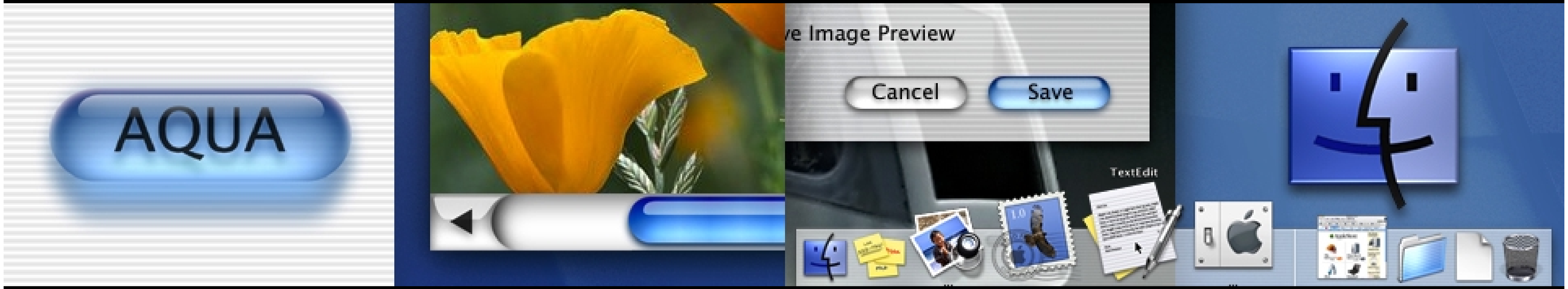
- Build smaller components, assemble into larger ones
- Is component to be reused across applications?
- How to manage state?
- Use WOB API validation
  - Design time only
- Provide default values for attributes

```
boolean canGetValueForBinding(String aBindingName)
boolean canSetValueForBinding(String aBindingName)
```

- Avoid namespace conflicts

Session 416

Q&A



Francois Jouaux, Manager, WebObjects Deployment
Kelly Kazem, Sr. Systems Engineer
Bob Frank, Consulting Engineer
Scott Sweet, Systems Engineer

Roadmap

417 Building Large-Scale Applications Practical development tips for developers	Room J2 **Fri., 2:00 p.m.**
915 WebObjects Feedback Forum A chance to give your feedback to us	Room J2 **Fri., 3:30 p.m.**

For More Information

http://www.apple.com/webobjects

http://enterprise.apple.com/wwdc2000

Visit the WebObjects lab downstairs!
Everyday from 11:00 a.m.–2:00 p.m.

Try out your WebObjects 4.5 Evaluation CD!

Who to Contact

Toni Trujillo Vian
Director, WebObjects Engineering
wofeedback@group.apple.com

Ernest Prabhakar

Product Line Manager, WebObjects
webobjects@group.apple.com

WWDC

Worldwide Developers Conference 2000

Think different.