Siebel Open UI Adv JavaScript API

Oracle Siebel CRM

DURATION	MODULES	COURSE CODE
3 Days	**6 Lectures**	**—**

Course Overview

This Siebel Open UI Advanced JavaScript API training shows you how to use the Siebel Open UI JavaScript (JS) application programming interface (API). Learn to implement common customizations to the application's user interface and functionality.

What You Will Learn

- 1 Course Introduction
- Lesson Agenda
- Instructor and Class Participants
- Training Site Information
- Course Audience
- Course Prerequisites
- Course Goal
- Course Objectives
- Course Methodology
- Course Materials
- Information Sources
- Course Agenda
- 2 Overview of Open UI Customization
- Objectives
- Siebel Open UI
- The Open UI Framework
- Siebel Open UI Client
- Parts of the Siebel Open UI Framework
- 1. Client Proxy
- 2. Presentation Models (PMs)
- 3. Physical Renderers (PRs)
- 3. Plug-in Wrappers
- 4. Cascading Style Sheets (CSS Files)
- Customizing the Open UI Framework
- Development Roles

- The JavaScript Application Programming Interface (JSAPI) for Open UI
- Lesson Highlights
- Practice
- 3 Administering the Manifest
- Objectives
- Review: The Siebel Open UI Manifest
- Review: Manifest Files
- Review: Manifest Expressions
- Review: Manifest Administration
- Default Behavior
- Extending Default Behavior
- Extending Existing Objects
- Expressions
- Create a New Conditional Expression
- Multiple Conditional Expressions
- Expression Groups
- Expressions within an Expression Group
- Review: Administer the Manifest
- 1. Identify the Name of the Object to Customize
- 2. Ensure the File to Apply is Registered
- 3. Select the Object to Customize
- 4. Associate the Object with a Condition
- 5. Associate the Object/Condition with a File
- 6. Apply the Modified Manifest
- 7. Test the Customization
- Lesson Highlights
- Practice
- 4 Using jQuery
- Objectives
- jQuery
- jQuery in JavaScript
- Selecting HTML Elements in jQuery
- Selecting Child HTML Elements in jQuery
- Performing Actions on the Selected Set
- Function Actions
- CSS Function Example
- Setting a Style Defined in a CSS Style Sheet
- html Function Example
- after/before Function Example
- Attaching an Event Handler Function to the Selected Set
- Example of a Click Event Handler
- Storing Selected Elements in a Variable
- Chaining Actions
- Choosing a jQuery Selector
- Disabling the Cache
- Caching versus Re-Reading the Manifest

- Use jQuery
- 1. Identify the Control or Field to be Manipulated
- iv
- 2. Add one or more jQuery selector/action statements
- 3. Test Your New Customization
- Lesson Highlights
- Practice
- 5 Customizing a Presentation Model for a Form Applet
- Objectives
- Review: The Presentation Model
- Review: File Format for a Presentation Model
- Presentation Model Templates
- Example Presentation Model Template File
- Customizing a PM with a New Method
- Using AddMethod() to Hook into a Framework Method
- Settings Arguments to AddMethod()
- Example: Define your Custom Method
- The Collection of Controls
- Finding Control Key and FieldName Values
- Methods Used in a PM to Work with Controls in an Applet
- Alternate Ways to Access a Single Control
- Example of Getting Controls: The ShowSelection() Method
- Accessing Controls in a Method Bound to ShowSelection()
- Accessing an Individual Field Value for ShowSelection()
- Customize a PM for a Form Applet
- Caveat: Client-Side Data Validation
- 1. Create the Basic PM file
- Use AddMethod() to bind a custom method to the appropriate internal
- framework method
- 3. Write a new method to implement the business requirement
- 4. Save the file with its new file name
- 5. Apply the New PM to the Applet
- 6. Test the Solution
- Lesson Highlights
- Practice
- 6 Customizing a Physical Renderer for a Form Applet
- Objectives
- Review: The Physical Renderer (PR)
- Review: File Format for a Physical Renderer
- Physical Renderer Templates
- Example Physical Renderer Template File
- Additional Physical Renderer Methods
- v
- Example Physical Renderer Template File
- The ShowUI() Method
- The BindData() Method

- The BindEvents() Method
- The EndLife() Method
- Custom Physical Renderers
- Some of the JavaScript API Methods Used in a PR
- Accessing a Specific Control/Field in the PR
- Customize a PR for a Form Applet
- 1. Create a Template PR File for a Form Applet
- 2. Modify the File to Achieve the Business Goal
- 3. Apply the New PR to the Applet
- 4. Test the Solution
- Lesson Highlights
- Practice
- 7 Presentation Model / Physical Renderer Interaction
- Objectives
- Separation of Concerns in the Open UI
- Advantages of SoC in Siebel Open UI
- PM / PR Interoperation
- Example of MVC Interoperation Using a Property in the PM Bound in the PR
- Methods Used to Create Interactions Between PR and PM
- Creating a Custom Property in the PM
- Changing the Custom Property in the PM
- Binding a Callback Method in the PR to a Property Change in the PM
- Accessing the Custom PM Property Value in the Callback Method in the PR
- Tips on Locating Code
- Tips on Locating Code: Details
- Create PM/PR Interaction
- 1. Add a Property to the PM
- 2. In the PM, Add Code to Set the Property As Needed
- 3. In the PR, Bind a Callback to the Property
- 4. In the PR, Code the Callback Method
- 5. Test the Results
- Lesson Highlights
- Practice
- 8 Debugging
- Objectives
- Sources of Problems in Scripts
- vi
- Verify that the Script was Loaded at Runtime
- Errors that Can Prevent Script Loading
- Check for Syntax Errors in the JavaScript that Prevent Execution from Starting
- Check for Runtime Exceptions that Halt Execution
- Search the Code for Logical Errors
- Debugging Tools
- Viewing Variable Values
- Inspecting the Call Stack
- Stepping Through Code

- Ensure that the Cache Is Disabled or Cleared
- Text Logging and Interactive Alerts
- Debug a Script
- 1. Verify that the Cache is Not Active
- 2. Verify that Script Has Been Loaded
- 3. Verify that There are No Syntax Errors in the JavaScript
- 4. Verify that There are No Runtime Errors in the JavaScript
- 5. Use Debugging Tools to Search for Logical Errors
- Lesson Highlights
- Practice
- 9 Customizing a List Applet
- Objectives
- Structure of a List Applet
- Placeholder
- Record Set
- An Individual Record
- Accessing an Individual Row by HTML ID
- Accessing a Field within a Row
- A Basic PR for a List Applet
- Search for a Row in a List Applet
- Search for a Field Value in a List Applet
- Alternate Looping Techniques
- BindData() Method in a List Applet
- ShowUI() Method
- A PM for List Applets
- Customize a List Applet
- 1: Create the Basic PR File
- 2: Add Code to the Appropriate Method
- 3: Administer the New PR
- 4: Test Your Changes
- vii
- Lesson Highlights
- Practice
- 10 Helper Techniques
- Objectives
- Helpers
- Additional Features
- Plug-in Wrappers
- Additional Methods of Plug-in Wrappers
- Example AttachPW
- Use a Plug-in Wrapper to Extend a Control on an Applet
- 1. Write the Plug-in Wrapper Code
- 2. Attach the Plug-in Wrapper
- 3. Administer the Plug-in Wrapper
- 4. Test the Results
- Additional Features

- The EventHelper() Object
- Implement an EventHelper() Object
- Example: Highlight a Field when the Mouse Enters it
- Working Example of Event Manager
- Additional Features
- HTML Template Manager
- Example: HTML Template Manager
- Working Example of HTML Template Manager
- Additional Features
- Responsive Web Pages
- Dynamic Layouts
- Default Widths
- Widths in Web Template Files
- Resulting Behavior
- Lesson Highlights
- Practice
- 11 Customization of Non-Applet Objects
- Objectives
- Applet and Non-Applet UI Elements
- postload
- Adding a Listener for postload
- Application-Level Customizations
- Example Global Customizations Appropriate for Use in Postload
- Making Applets Collapsible
- viii
- Sorting in jQuery
- A Custom Sorting Function
- Customize a Non-Applet Element using postload
- 1. Create a Custom File with a Listener for the postload Event
- 2. Write Custom Code in the Event Handler
- 3. Administer the File
- 4. Test your Changes
- Lesson Highlights
- Practice
- 12 Using External Libraries and Web Sites
- Objectives
- Using External Code
- External jQuery Libraries
- Using External jQuery Libraries
- Copy the External Package to Your Site
- Include the External jQuery Package as a Dependency in Your Code
- Follow the Documentation for the External Package
- JavaScript Code Accessed via the Web
- Accessing Code from a URL Asynchronously
- Steps to Access Code Online via a URL
- Asynchronously Load the API

- Load Module/Packages with a Callback
- Check that Code Is Available
- Accessing a Web-Based Application
- Calling an External Web Application
- Steps to Open an External Web Application in a New Window Based on a User
- Action in the UI
- 1. Create a String with an Appropriate URL
- 2. Populate the URL from Values from the Applet
- 3. Write JavaScript to Open a Window to the URL
- 4. Call JavaScript When the User Action is Detected in the UI
- Creating a New Link Dynamically
- Access an External Web Site with Data from the Siebel Application
- 1. Create a Presentation Model (PM)
- 2. Add a Property and Methods to the PM
- 3. Create a Physical Renderer (PR)
- 4. Add HTML that will be Used as a Link
- 5. Add a Binding
- 6. Apply the New PR / PM
- 7. Test the New Functionality
- ix
- Lesson Highlights
- Practice
- 13 Siebel Business Services and Profile Attributes
- Objectives
- Siebel Business Service
- Accessing a Siebel Business Service
- Make the Service Available to the Application
- Write Code to Access the Service
- Solution Steps
- 1. Identify the Service, its Methods, and the Parameters for Each Method
- 2. If Necessary, Configure the Application to Use the Business Service
- 3. Write Code for the Business Requirement
- 3. Write Code in a Custom Postload Handler
- 4. Apply the New Code to the Application
- 5. Test Your Change
- Lesson Highlights
- Practice
- 14 Using Siebel Tools with Siebel Open UI
- Objectives
- Add Presentation Model Properties to a View, Applet, or Control
- Presentation Model Properties as User Properties
- Read Presentation Model Properties in the Presentation Model Code
- 1. Load the Siebel Constants
- 2. Get the Presentation Model Properties
- 3. Parse the Property Set
- Complete Code Example: Applet

- Result
- Special Property: EnableDragAndDropInList
- Enable Drag-and-Drop Imports
- 1. Add the User Property to the Applet
- 2. Prepare the Spreadsheet
- 3. Test the Results
- Lesson Highlights
- Practice
- 15 User Preferences
- Objectives
- User Visualization
- User Visualization in Siebel Open UI
- x
- Selecting Default User Visualization
- Configuring Visualizable Applets
- Visualizable Web Template
- Physical Renderer and Presentation Model
- Create a Visualization Model
- 1. Add the Web Templates to the Applet
- 2. Edit the Web Layouts
- 3. Administer the Web Templates
- 4. Administer the Manifest
- 5. Test the Results
- Additional Visualizations
- Other User Preferences
- User Preferences
- Read a User Preference
- Set a User Preference
- Example: A Button that Hides and Shows Applets
- 1. Add the Button to the User Interface
- Verify the Button
- 2. Add the Script that Reacts to the Button
- 3. Test the Results
- Lesson Highlights
- Practice
- 16 Deploying to a Production Environment
- Objectives
- Technical Challenge
- Technical Solution: Migrate Customizations to the Server
- 1. Use Standard Techniques to Migrate Siebel Repository Changes
- 2. Copy the Files
- File Considerations
- 3. Administer the Manifest for the Enterprise
- Manifest Considerations
- 4. Test the Results
- Lesson Highlights
- Practice