

Adobe Flex 3: リッチクライアントアプリケーション開発 基礎講座

- Adobe Flex 3 の概要
 - リッチインターネットアプリケーションについて
 - アプリケーションの進化
 - RIA の特徴
 - RIA の利点
 - RIA 開発用の Adobe ツール
 - Adobe Flex について
 - Adobe Flex SDK
 - Adobe Flex Builder について
 - Adobe Flex Builder Standard
 - Adobe Flex Builder Professional
 - RIA の基盤技術である Flash Player / Flash Virtual Machine について
 - Flex アプリケーションのプロセスフローについて
 - Flex アプリケーションからのリモートデータリソースへのアクセス
 - ヘルプおよびその他のリソースへのアクセス
 - ドキュメンテーション
 - Flex URL
 - BLOG
- Flex Builder ファーストステップガイド
 - Eclipse と Flex Builder の関係について
 - Eclipse Foundation
 - Flex Builder
 - Flex Builder インターフェース
 - エディタの概要
 - ビューの概要
 - パースペクティブの概要
 - プロジェクトの作成およびアプリケーションの作成の基本
 - プロジェクトの作成
 - メインアプリケーションファイルの使用
 - アプリケーションファイルの実行
 - アプリケーションのコンパイル

- ワークスペースによるプロジェクトのグループ分け
 - ウォークスルー1: メインアプリケーションファイルの作成と実行
 - Flex アプリケーションのデバッグ
 - 変数値のトレース
 - ブレークポイントの設定
 - アプリケーションのデバッグ
 - ウォークスルー2: デバッグおよびデバッグパースペクティブの使用
 - Adobe Flex Builder でのヘルプへのアクセス
 - Flex Builder 内のヘルプ
 - ドキュメンテーション
- Flex の基本
 - シンプルな Flex アプリケーションの作成
 - 名前空間の指定
 - レイアウトプロパティの設定
 - UI コントロールの追加
 - コンポーネントプロパティの指定
 - MXML コードへのコメント
 - ウォークスルー1: 初めての Flex アプリケーション作成
 - 画像の表示
 - 画像を表示する方法
 - SVG (Scalable Vector Graphic) ファイルでの作業の制限
 - ウォークスルー2: アプリケーションへの画像の追加
 - コンテナを使用した Flex アプリケーションのレイアウト
 - Flex Component Explorer でのさまざまなタイプのコンテナの表示
 - コンテナの機能
 - レストランアプリケーションの例
 - Box コンテナ
 - Canvas コンテナ
 - <mx:Application>タグでのレイアウトの使用
 - ウォークスルー3: 基本レイアウトコンテナの使用
 - Panel コンテナの使用
 - Panel タグの使用
 - ウォークスルー4: アプリケーションモジュールのパネルへの分割
 - ControlBar コンテナの使用
 - ウォークスルー5: ControlBar コンテナの使用

- ユーザーインターフェースコントロールの追加
 - Label コントロールについての再考
 - TextInput コントロール
- コンポーネント間のデータバインディング
 - インスタンス名の割り当て
 - データバインディングの作成
- ウォークスルー6: データバインディングの作成
- カスタム MXML コンポーネントでのアプリケーション構築
 - カスタム MXML コンポーネントの使用
 - カスタム MXML コンポーネントの作成
 - カスタム MXML コンポーネントのインスタンス化
- ウォークスルー7: カスタム MXML コンポーネントの作成とインスタンス化
- カスタム MXML コンポーネントのプロパティおよびメソッドの作成
 - カスタム MXML コンポーネント内でのプロパティの作成
 - カスタム MXML コンポーネントプロパティの参照
 - バインド可能プロパティの作成
 - カスタム MXML コンポーネントのメソッドの作成
 - カスタム MXML コンポーネントメソッドの参照
 - ActionScript コードへのコメント
- ウォークスルー8: コンポーネントでのプロパティおよびメソッドの作成
- イベントの処理
 - イベントについて
 - システムイベント
 - ユーザーイベント
 - インライン ActionScript を使用したイベントハンドラの作成
 - インライン ActionScript を使用したイベントハンドラの作成
 - MXML タグでの ActionScript の使用
 - ウォークスルー1: インライン ActionScript の使用
 - ActionScript 関数内でのイベント処理
 - MXML ドキュメントでの関数の定義
 - ウォークスルー2: イベントハンドラ用の関数の使用
 - 外部ファイルへの ActionScript 関数の記述
 - イベントオブジェクトについて
 - イベントオブジェクトのプロパティ
 - event オブジェクトのデータ型指定

- currentTarget と target
 - コード例
 - Image クラスにおける特別なケース
 - ウォークスルー3: イベントオブジェクトについて
 - addEventListener()メソッドの使用
 - ウォークスルー4: addEventListener()メソッドの使用
- 制約ベースのレイアウトを使用したアプリケーションのレイアウト
 - 絶対配置について
 - 絶対配置用のコンテナサポート
 - 絶対配置の特徴
 - Canvas コンテナ内でのコンポーネントの配置
 - Canvas コンテナの長所と短所
 - ウォークスルー1: Canvas 内でのコンポーネントの配置
 - Flex Builder を使用した制約ベースのレイアウトの設定
 - 制約ベースレイアウトと従来のコンテナレイアウト
 - 視覚的なアンカーの指定
 - ウォークスルー2: デザインモードを使用した制約ベースのレイアウトの実装
 - MXML での制約ベースのレイアウトの設定
 - horizontalCenter および verticalCenter の使用
 - ウォークスルー3: MXML での制約ベースのレイアウトの実装
 - 高度な制約の使用
 - ConstraintRow および ConstraintColumn の使用
 - ネストされたコンテナ内での制約ベースのレイアウトの使用
 - ウォークスルー4: 制約ベースのレイアウトでのカスタムコンポーネントの使用
- ラボ1
 - 基本的なフォトギャラリーアプリケーションとホームページの作成
 - Contribute ページコンポーネントの作成
 - Gallery ページコンポーネントの作成
- アプリケーションデザインでのビューステートの使用
 - ビューステートについて
 - ビューステートについて
 - ビューステートの作成
 - ビューステートを使用する利点
 - デザインモードでのビューステートの作成

- ウォークスルー1: Contact ページの2つのステートの作成
- ビューステートのコントロール
- ウォークスルー2: Contact フォームでのステートの切り替え
- 生成された MXML コードの確認
 - AddChild タグの使用
 - RemoveChild タグの使用
 - SetProperty タグの使用
 - SetEventHandler タグの使用
- ウォークスルー3: MXML を使用したビューステートの実装
- カスタムコンポーネントを含むビューステートの作成
- ウォークスルー4: カスタムコンポーネントでのビューステートの使用
- アプリケーションナビゲーションの作成
 - ナビゲータコンテナとコントロールについて
 - LinkBar コントロールの使用
 - TabBar コントロールの使用
 - ViewStack コンテナの使用
 - アクティブな ViewStack 子コンテナの設定
 - ViewStack コンテナのコンテンツを管理するための LinkBar コンテナと TabBar コンテナの再考
 - ViewStack の子コンテナの表示とサイズ変更
 - 動的なボタンの有効化
 - ウォークスルー1: ViewStack コンテナおよび TabBar コントロールを使用した Café Townsend アプリケーションのナビゲート
 - TabNavigator コンテナの使用
 - Accordion コンテナの使用
 - ウォークスルー2: TabNavigator および Accordion コンテナを使用した Café Townsend アプリケーションのナビゲート
 - ButtonBar & ToggleButtonBar の使用
 - ApplicationControlBar コンテナの使用
 - ウォークスルー3: Café Townsend アプリケーションへの ApplicationControlBar コンテナの追加
- アプリケーションのカスタマイズ
 - Flex アプリケーションのルックアンドフィールのカスタマイズ
 - Flex コントロールスタイルのデフォルトの変更
 - アニメーションエフェクトを使用したユーザーとのインタラクション

- ルックアンドフィールを変更するためのスタイルの変更
 - MXML コンポーネント属性を使用したインラインスタイルの設定
 - `setStyle()`メソッドを使用した ActionScript での個々のコンポーネントのスタイル設定
 - ActionScript と CSS スタイルについて
 - CSS(カスケーディングスタイルシート)を使用したスタイル設定
 - ボタンステートのスタイル
- テーマの使用
- ウォークスルー1: Café Townsend アプリケーションのルックアンドフィールの変更
- コンポーネントへのビヘイビアの適用
 - ビヘイビアの基本
 - 使用できるエフェクトの確認
 - パラレルおよびシーケンスエフェクトの作成
- ウォークスルー2: Café Townsend 座席図へのビヘイビアの追加
- ビューステートへのトランジションの適用
- ウォークスルー3: ステートトランジション中の Café Townsend 問い合わせパネルのサイズ変更
- ラボ2
 - アプリケーションナビゲーションとカスタムコンポーネントの追加
 - デザインモードでのビューステートの作成とトランジション
 - MXML でのビューステートの作成
 - fStop アプリケーションの外観の変更
- ActionScript データモデルの使用
 - MVC デザインパターンの使用
 - MVC エlement
 - MVC の利点
 - ラボアプリケーションの例
 - MXML データモデルの作成
 - MXML データモデルの作成
 - データモデルとしての ActionScript クラスの使用
 - クラスパスおよびパッケージ
 - クラスファイルの作成
 - クラスの使用
 - クラスコンストラクタの作成
 - クラスプロパティの定義

- MXML での ActionScript クラスのインスタンス化
 - プロパティをバインド可能にする
 - ウォークスルー1: ActionScript クラスの MXML によるインスタンス化
 - パラメータを持つ ActionScript コンストラクタの作成
 - This キーワードの使用の詳細
 - コンストラクタパラメータの使用
 - ウォークスルー2: ActionScript クラスの ActionScript によるインスタンス化
 - クラスメソッドの定義
 - メソッドの呼出
 - 静的メソッドの定義
 - 静的メソッドの呼出
 - ウォークスルー3: ActionScript クラスへのメソッドの追加
- カスタムイベントを使用したコンポーネント間のデータ交換
 - バインドを使用した場合の問題点の理解
 - バインド使用の問題点
 - 疎結合コンポーネントの作成
 - カスタムイベントの作成
 - イベントの定義
 - イベントオブジェクトの作成
 - イベントの送出
 - 例
 - イベントを処理するイベントハンドラの実装
 - ウォークスルー1: カスタムイベントの作成、送出、処理
 - カスタムイベントでのデータ送信
 - カスタムイベントクラスの必要性の確認
 - カスタムイベントオブジェクトの必須作業の確認
 - カスタムイベントクラスの作成
 - プロパティの追加
 - コンストラクタの作成
 - Clone メソッドのオーバーライド
 - カスタムイベントクラスの使用
 - カスタムイベントの送出
 - 例
 - カスタムイベントの処理
 - ウォークスルー2: カスタムイベントオブジェクトの作成および送出

- データエントリーフォームの作成
 - Form コンテナの使用
 - フォームコンポーネントの使用
 - FormItem コンテナの使用
 - Form カスタムコンポーネントの作成
 - ウォークスルー1: データエントリーフォームコンポーネントの作成
 - フォームデータの共有
 - 値オブジェクトの作成
 - カスタムイベントの作成
 - カスタムイベントの宣言と送出
 - カスタムイベントの処理
 - ウォークスルー2: アプリケーションでのフォームデータの共有
 - フォームデータの検証
 - MXML での Validator の作成
 - ActionScript でのバリデータの作成
 - 必須値の処理
 - イベントでの検証のトリガ
 - 検証が失敗した場合
 - Number Validator クラスの使用
 - ウォークスルー3: 数値入力の検証
 - ActionScript での検証のトリガ
 - 自動検証の無効化
 - Validate()メソッドの使用
 - 複数の検証のトリガ
 - ウォークスルー4: ActionScript での検証のトリガ
- HTTPService での XML データの取得
 - 実行時の XML データの取得
 - HTTPService クラスの使用
 - HTTP リクエストを行うプロセス
 - 結果の処理
 - 結果が ArrayCollection に解析される場合
 - データバインディングでの結果の使用
 - ウォークスルー1: HTTPService による実行時のデータ取得
 - イベントハンドラを使用した結果の処理
 - 例
 - ウォークスルー2: result イベントの使用

- エラー処理
 - Alert ポップアップでのメッセージの表示
- ウォークスルー3: エラーイベントの処理
- さまざまなドメインへの HTTP リクエスト
 - クロスドメインポリシーファイルの使用
- ウォークスルー4: クロスドメインポリシーのテスト
- パラメータでの HTTP リクエスト
 - 明示的にパラメータを渡す方法の使用
 - パラメータのバインディングの使用
- ウォークスルー5: パラメータをとまなう HTTPService の使用
- DataGrid を使用したデータの表示
 - DataGrid コントロールの使用
 - DataGrid へのデータの供給
 - DataGrid 列の指定
 - ウォークスルー1: DataGrid 列の指定
 - DataGrid 列のフォーマット
 - 再利用可能なラベル関数の作成
 - ウォークスルー2: DataGrid 列のデータフォーマット
 - アイテムレンダラーとアイテムエディタの使用
 - アイテムレンダラーとアイテムエディタ
 - アイテムレンダラーとアイテムエディタの種類
 - ウォークスルー3: ドロップインアイテムエディタの使用
 - インラインアイテムレンダラーとインラインアイテムエディタ
 - ウォークスルー4: インラインアイテムエディタの使用
 - アイテムレンダラーとアイテムエディタのコンポーネント
 - ウォークスルー5: コンポーネントアイテムレンダラーの使用
 - TileList および HorizontalList の使用
 - アイテムレンダラーの使用
 - ウォークスルー6: TileList および HorizontalList でのデータ表示
 - リストベースコンポーネントでのイベントと選択されたアイテムの使用
 - ウォークスルー7: TileList での change イベントの使用
- ラボ3
 - リモートデータの取得とデータ構造の作成
 - ギャラリーでの動的画像の表示
 - ビューステートでのクリックされた画像の表示
 - 購入された写真の写真情報を渡す

- ショッピングカートでの購入された社員の表示