



Adobe Flex 3

اساسيات البرمجة بالفليكس

محمد محمود ابراهيم موسى

WWW.SCABTECH.COM | MOHAMMED.AAU@GMAIL.COM

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

شكراً وتقدير:

الحمد لله و الصلاة و السلام على رسول الله .. اللهم لك الحمد و الشكر و الثناء على توفيقك و تيسيرك سبل الخير لنا ، يسعدني ان اتقدم بخالص شكري الي كل من علمني درسا في حياتي ، الي كل من قدم لي نصحاً او نقداً في حياتي ، لكم مني كل الشكر والتقدير والاحترام.

اهـداً:

يسعدني ويشرفني ان اهدي هذا العمل المتواضع الي طلابي بجامعة الزعيم الازهري كلية علوم الحاسوب وتقانة المعلومات ، اعضاء فريق مشروع احلام ، شبكة ازكا التعليمية.

تمهيداً:

بين يديكم كتاب اساسيات البرمجة بالفليكس ، وهو كتاب يشرح ببساطة مفاهيم البرمجة باستخدام بيئة التطوير Adobe Flex (Adobe Flash Builder) ، حيث تعتبر الفليكس من بيئات التطوير الحديثة في عالم البرمجة والتصميم ، وبيئة التطوير فليكس العديد من المزايا حيث تسمح هذه البيئة بانشاء تطبيقات الويب وتطبيقات سطح المكتب وفي الاصدارات الحديثة من فليكس يمكن انشاء تطبيقات الهاتف المحمول (حيث تدعم فليكس حتى الان منصات Android ، IOS و Blackberry).

المحتويات:

2	المحتويات:
3	الفصل الاول: مقدمة
4	صفحات HTML وصفحات الانترنت التفاعلية:
5	الفليكس (Flex):
6	بيئة التطوير Adobe Flex Builder 3:
14	الفصل الثاني: إنشاء مشروع بلغة الفليكس
27	الفصل الثالث: لغة ActionScript
27	مفهوم ال Class:
35	التعليقات Comments:
35	الدوال Function:
37	تمرير الوسائط Passing Parameters:
39	التعامل مع الاحداث Handling Events:
41	استخدام الوسم [Bindable]:
42	التفاعل مع المستخدم:
43	محددات الوصول Access Modifiers:
44	إعادة الهيكلة Refactoring:
46	الفصل الرابع: وعاء (حاوية) التطبيق (Application Container)
46	مدير التنسيق (Layout Manager):
49	حاويات التنسيق Layout Containers:
50	HBox و VBox:
54	Form Container:
57	Panel Container:
61	Navigation Container:
62	الحاوية ViewStack Container:
70	عمل Navigation باستخدام ActionScript:
71	الحاوية TabNavigator والحاوية Accordion:
74	الحالات States:
83	الحالات والتعليمات البرمجية:
84	التمرير والحالات Rollovers and Stats:
84	استيراد الموارد Resources الجاهزة الي المشروع:
96	الفصل الخامس: الاحداث والمكونات Events and Components
96	الاحداث Events:
99	كانن الحدث The Event Object:
100	الدالة AddEventListener:
103	المكونات Components:
110	المكونات والبيانات Components and Data:
113	الاحداث المخصصة Custom Events:
119	تمرير البيانات Passing Data:
122	المراجع:

الفصل الاول: مقدمة

الحمد لله الواحد المعبود، عم بحكمته الوجود، وشملت رحمته كل موجود، أحمده سبحانه وأشكره وهو بكل لسان محمود، وأشهد أن لا إله إلا الله وحده لا شريك له الغفور الودود، وعد من أطاعه بالعزة والخلود، وتوعد من عصاه بالنار ذات الوقود، وأشهد أن نبينا محمداً عبد الله ورسوله، صاحب المقام المحمود، واللواء المعقود، والحوض المورود، صلى الله عليه وعلى آله وأصحابه، الركع السجود، والتابعين ومن تبعهم من المؤمنين الشهود، وسلم تسليماً كثيراً إلى اليوم الموعود.

يعتبر هذا الكتاب مدخل الي البرمجة باستخدام بيئة التطوير Adobe Flex، وتعتبر الفليكس من بيئات التطوير الحديثة في عالم البرمجة والتصميم، ولبينة التطوير فليكس العديد من المزايا حيث تسمح هذه البيئة بإنشاء تطبيقات الويب وتطبيقات سطح المكتب وفي الاصدارات الحديثة من فليكس يمكن انشاء تطبيقات الهاتف المحمول (حيث تدعم فليكس حتى الان منصات Android، IOS، و BlackBerry)، كل الامثلة والتطبيقات الواردة في هذا الكتاب تمت باستخدام بيئة التطوير Adobe Flex الاصدار الثالث، لذلك سيجد مستخدمي الاصدارات الاحدث (الاصدار الرابع) من بيئة التطوير Adobe Flex اختلافات بسيطة.

في هذا الكتاب نركز على فهم مبادئ البرمجة باستخدام الفليكس، تم تقسيم الكتاب الي خمسة فصول تتناول المفاهيم الاساسية لبيئة التطوير فليكس، في الفصل الاول سنتعرف على مبداء البرمجة باستخدام بيئة التطوير فليكس، وفي الفصل الثاني سنتعلم كيفية انشاء المشاريع والتطبيقات باستخدام الفليكس، في الفصل الثالث سنقدم شرح مختصر للغة البرمجة Action Script بالاضافة الي بعض مفاهيم البرمجة كائنية المنحى، الفصل الرابع نشرح فيه انواع حاويات التطبيقات، الفصل الخامس سنتعرف على كيفية التعامل مع المكونات والاحداث.

الانترنت:

قبل أن نبدأ الحديث انشاء التطبيقات باستخدام فليكس (Flex) نحن بحاجة إلى فهم ودراسة تاريخ الانترنت بصورة عامة، ونحن نتحدث عن تاريخ لان التقنيات المختلفة التي نراها اليوم مرت بمراحل زمنية متعددة مع تطور الانترنت، سنتحدث عن هذه التقنيات في مقدمة بسيطة لأنه من المهم ان نفهم هذا التطور لنرى ماذا قدمت فليكس.

صفحات HTML وصفحات الانترنت التفاعلية:

مواقع الانترنت في السابق كانت تحتوي على نصوص فقط مع بعض الروابط التي تسمح لك بالانتقال إلى صفحات أخرى ونسبة لان الانترنت كان بطيء جدا فلم تكن صفحات الانترنت تحتوي على الكثير من الصور والرسومات، يتم تصميم الصفحات بلغة HTML والصفحات التي يتم تصميمها بلغة HTML هي صفحات ثابتة غير متغير بمعنى إن هذه الصفحات محتواها لا يتغير إلا إذا قام شخص ما بفتح كود HTML وقام بإجراء التغير عليه.

عندما نقوم بكتابة عنوان كهذا في متصفح الانترنت www.aau.edu.sd في متصفح الانترنت فان هذا العنوان سيتم إرساله عبر سلسلة من الموجهات (Routers) على شبكة الانترنت العالمية (www) حتى يصل إلى خادم الانترنت المضيف لهذا الموقع يقوم الخادم بالبحث عن صفحة الـ HTML ثم يقوم بإرجاعها إلى متصفح الانترنت الخاص بالعميل (Client)، في هذا الكتاب لن نتحدث عن بناء صفحات HTML، فقط سنركز على مفاهيم برمجة تطبيقات الانترنت باستخدام الفليكس.

ظهرت العديد من التقنيات لبنا صفحات الانترنت التفاعلية أي أن محتوى هذه الصفحات متغير، هناك العديد من التقنيات التي تستخدم في معالجة صفحات الانترنت التفاعلية او الديناميكية ومن أشهر هذه التقنيات:

• **CFM:** ColdFusion

• **ASP:** Classic Microsoft Active Server Pages

• **ASPX:** Microsoft .NET Active Server Pages

• **JSP:** Java Server Pages

• **PHP:** A Hypertext preprocessing

كل هذه التقنيات تقوم بأداء نفس المهام مع الاختلاف في درجة السهولة والتعقيد، حيث تقوم باستقبال الأوامر من خادم الانترنت وإرسالها إلى خادم قاعدة البيانات باستخدام تعليمات SQL.

الفليكس (Flex):

كما لاحظنا إن في صفحات الانترنت العادية الانتقال من صفحة إلى أخرى يتطلب إرسال طلب الصفحة إلى الخادم في كل مرة، حيث تتم عملية البحث عن الصفحة، أما في الصفحات التفاعلية فان خادم الانترنت يأخذ الطلب ويرسله إلى واحدة من تطبيقات الخادم المختلفة التي قمنا بذكرها سابقا (asp، PHP،) والذي بدوره يقوم بإرسال الطلب إلى خادم قواعد البيانات ثم يتم تجميع البيانات ويتم كتابة صفحة HTML جديدة لعرض البيانات ثم إرسالها إلى متصفح الانترنت الخاص بالعميل.

قامت شركة Macromedia (الآن Adobe) التي قدمت Flash MX بتقديم تقنية جديدة Rich Internet Application (RIA) معتمدة على تقنية الفلاش Flash وتقلبت هذه التقنية على كثير من قيود صفحات HTML التقليدية وتتميز هذه التقنية في أن تطبيقات الانترنت تقريبا لا يمكن تمييزها عن تطبيقات سطح المكتب.

تطبيقات RIA لا تتطلب إعادة بناء الصفحات بالكامل وإنما يتم استرجاع البيانات المطلوبة فقط وهذا قلل من حجم الطلبات المرسله إلى الخادم.

بعد أن أطلقت Macromedia برنامج Flash MX قدمت لغة الـ ActionScript 2.0 كتحديث للغة الإجرائية ActionScript 1.0 للإيفاء باحتياجات تطبيقات RIA ، لغة ActionScript 2.0 كانت لغة شبه كائنيه المنحى .

اشتكى العديد من المبرمجين من هذه اللغة في أن أدوات التصحيح (Debugging Tools) كانت غير موجودة بالكامل، كما أن العديد من المطورين اشتكى من أن تطوير تطبيقات RIA على بيئة الفلاش كانت معقدة.

لمعالجة هذه القضايا الكثيرة قدمت شركة Macromedia في عام 2004 تقنية الفليكس Flex، حيث أعطت هذه التقنية المطورين بيئة تقليدية للبرمجة بدون تعقيدات التصميم الكثيرة الموجود في الـ Flash، تتميز هذه التقنية بان لها أدوات لتطوير تطبيقات RIA شبيه ببرنامج Dreamweaver تسمى Flex Builder لكنها لم تحصل على شعبية كبيرة نسبة لقيود ActionScript 2.0 وعدم دعمها الكامل للـ OOP.

تم إطلاق الإصدار الثاني من فليكس في عام 2006 والتحديث لم يكن في أصل الفليكس فحسب بل شمل الإعلان عن لغة ActionScript 3.0 والتي تدعم البرمجة كائنيه المنحى بشكل كامل.

بيئة التطوير 3 Adobe Flex Builder:

Adobe Flex Builder 3 هي عبارة عن بيئة متكاملة لتطوير تطبيقات الفليكس، يتكون Flex Builder 3 من ثلاث مكونات اساسية:

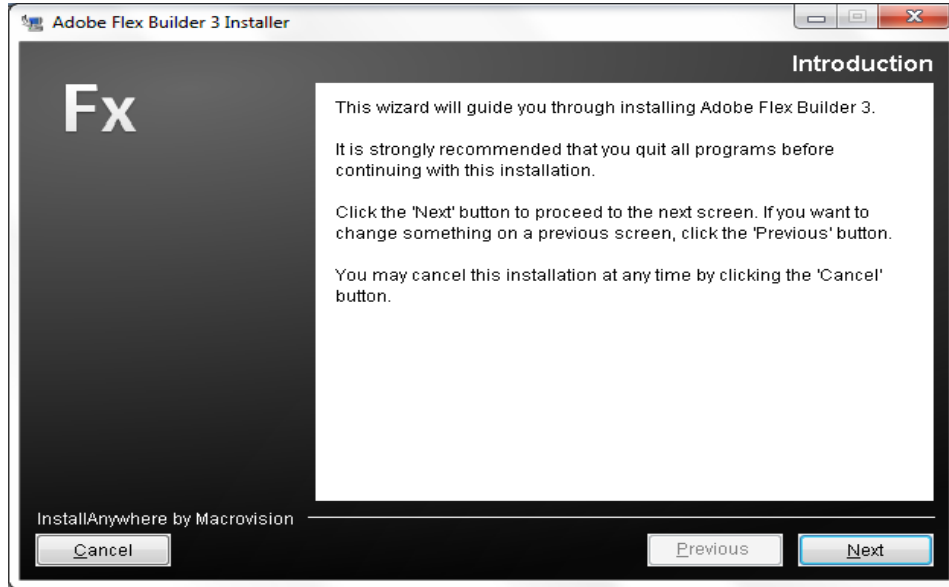
- **Flex SDK** : هي مجموعة من فئات لغة ActionScript 3.0 اللازمة لبناء وتشغيل ونشر تطبيقات الفليكس Flex
- **The Eclipse plug-in**: هذه المكونات تساعد في بناء تطبيقات فليكس Flex باستخدام بيئة التطوير Eclipse.
- **Flash Player 9**: يتم تشغيل تطبيقات الفليكس بواسطة مشغل 9 flash player أو أي إصدار احدث.

بناء على نظام التشغيل المستخدم يمكن تثبيت Adobe Flex Builder 3 من الاسطوانة أو تحميله بصورة مباشرة من الانترنت حيث تبدأ عملية التثبيت ببرنامج InstallAnyWhere يأخذ تثبيت هذا البرنامج ما لا يزيد عن دقيقتين ، أول نافذة تظهر تطلب منك تحديد لغة التثبيت.



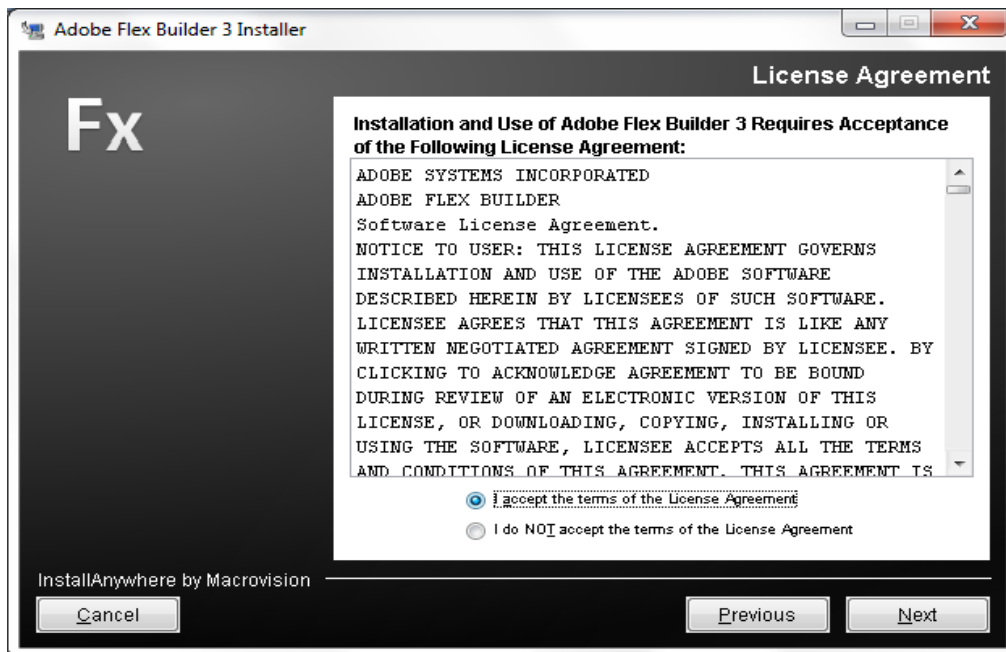
شكل 1-1 تحديد لغة التثبيت

من الأفضل إغلاق كل البرامج والنوافذ أثناء عملية التثبيت خاصة المتصفحات لان بيئة تطوير التطبيقات Flex Builder 3 ستقوم بتثبيت مشغل الفلاش الخاص بها على المتصفح وهو Adobe Flash Player 9 النافذة التالية توضح ذلك.



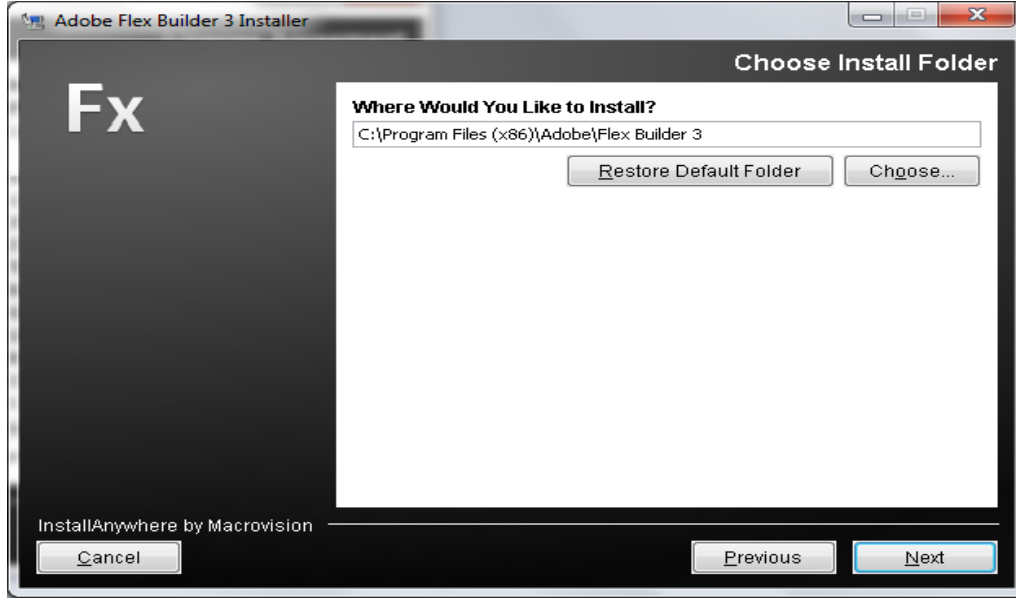
شكل 1-2 هذه النافذة توضح بعض الإرشادات لإكمال عملية التثبيت

النافذة التالية توضح شروط ترخيص البرنامج ويجب الموافقة عليها لاستمرار عملية التثبيت



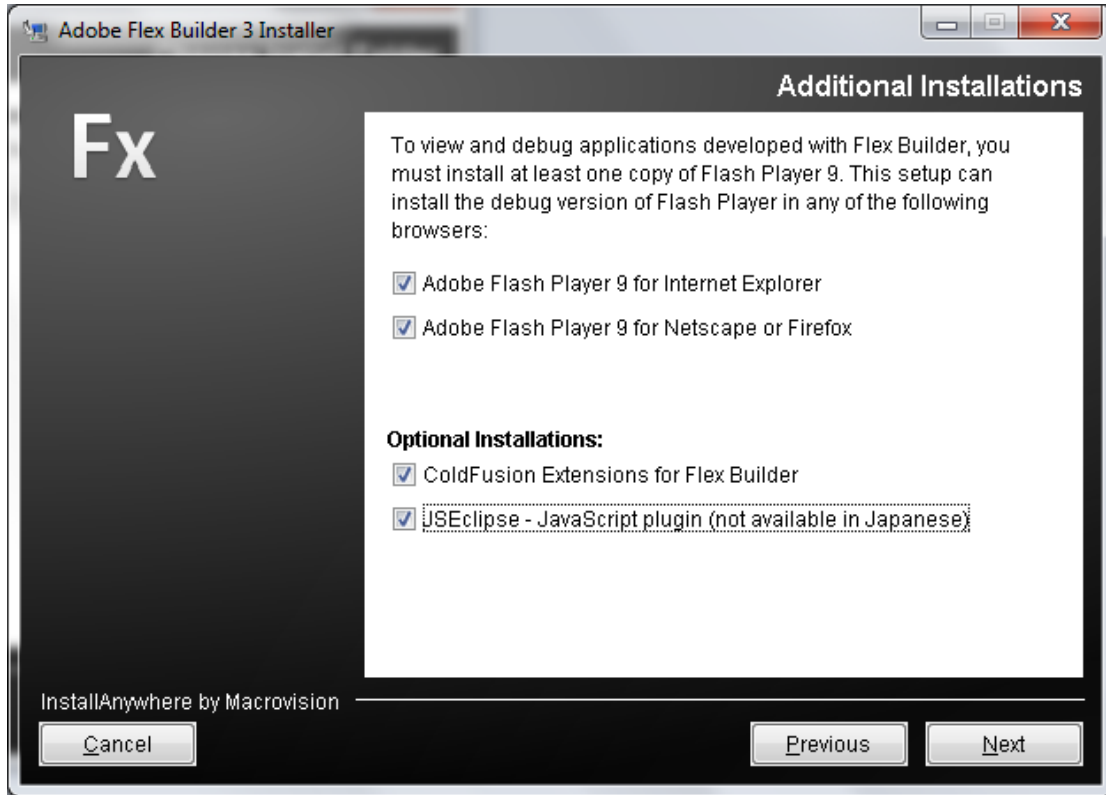
شكل 3-1 شاشة شروط ترخيص البرنامج

المرحلة التالية من عملية التثبيت هي تحديد المجلد الذي سيتم فيه تثبيت البرنامج والملفات المتعلقة بالبرنامج، إذا لم يكن هناك سبب اترك المجلد الافتراضي للتثبيت واضغط Next.



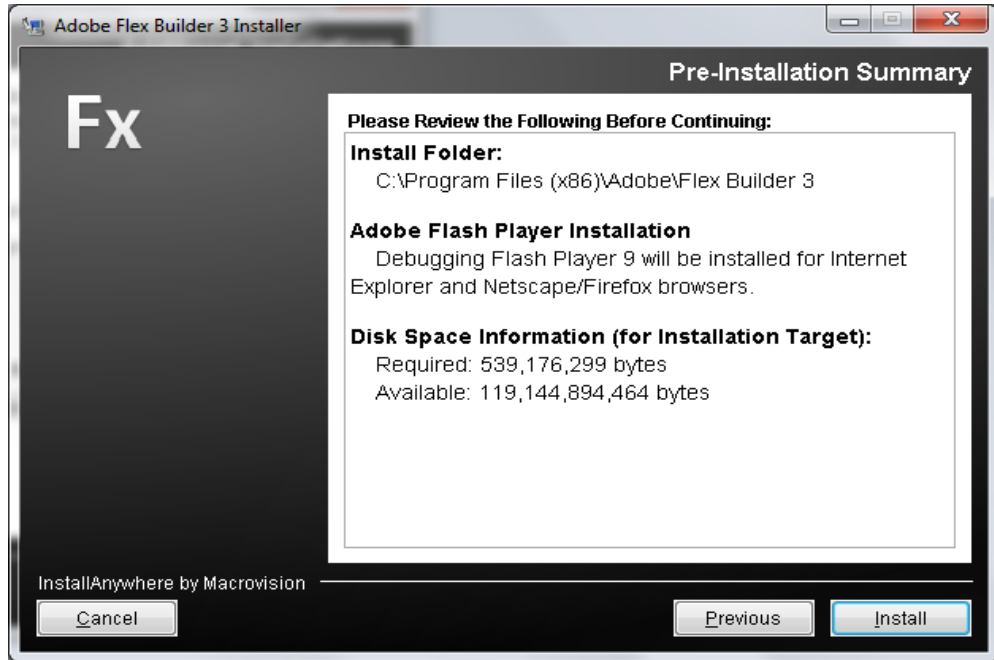
شكل 4-1 نافذة تحديد المجلد الذي يتم فيه تثبيت البرنامج

المرحلة التالية من التثبيت مهمة جدا حيث يتم تثبيت مشغل الفلاش على كل المتصفحات المثبتة على الكمبيوتر مشغل الفلاش له الإمكانية على معالجة ملفات الـ SWF التي يتم إنشائها في الـ Flex كما لك حرية الاختيار في تثبيت الإضافات على الـ Eclipse إذا كنت ترغب في البرمجة باستخدام JavaScript أو لغة الـ ColdFusion



شكل 5-1 تثبيت Adobe Flash Player

النافذة الأخير قبل بدء عملية التثبيت تسمح لك بمراجعة إعداد عملية التثبيت مثل موقع ملفات البرنامج ومشغل الفلاش والإضافات التي تم اختيارها بعد التحقق من كافة الإعدادات اضغط على Install لتبدأ في عملية التثبيت.

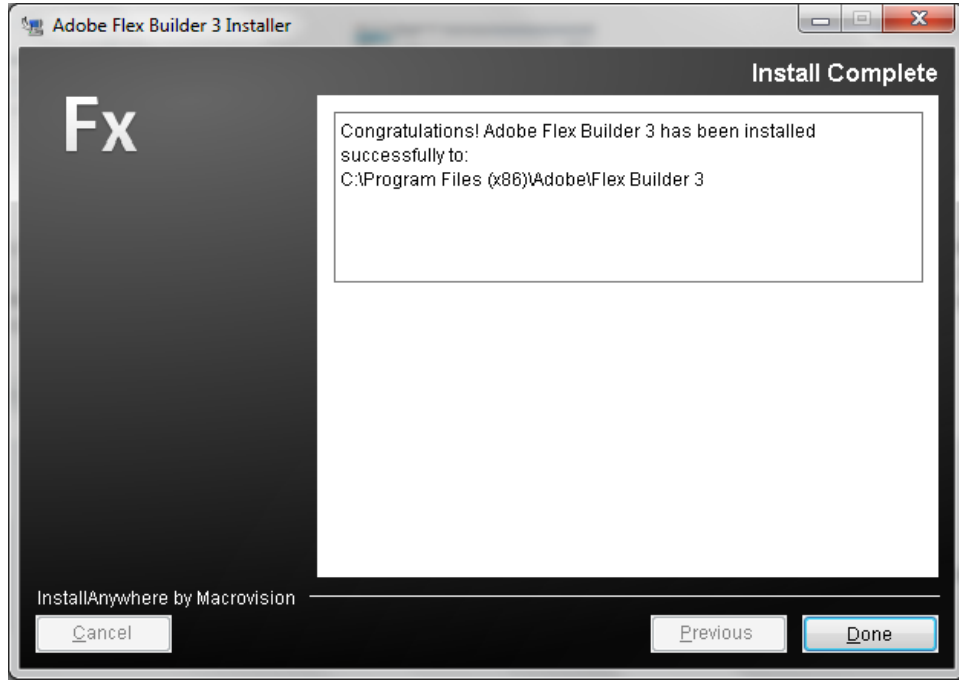


شكل 6-1 نافذة مراجعة إعدادات التثبيت



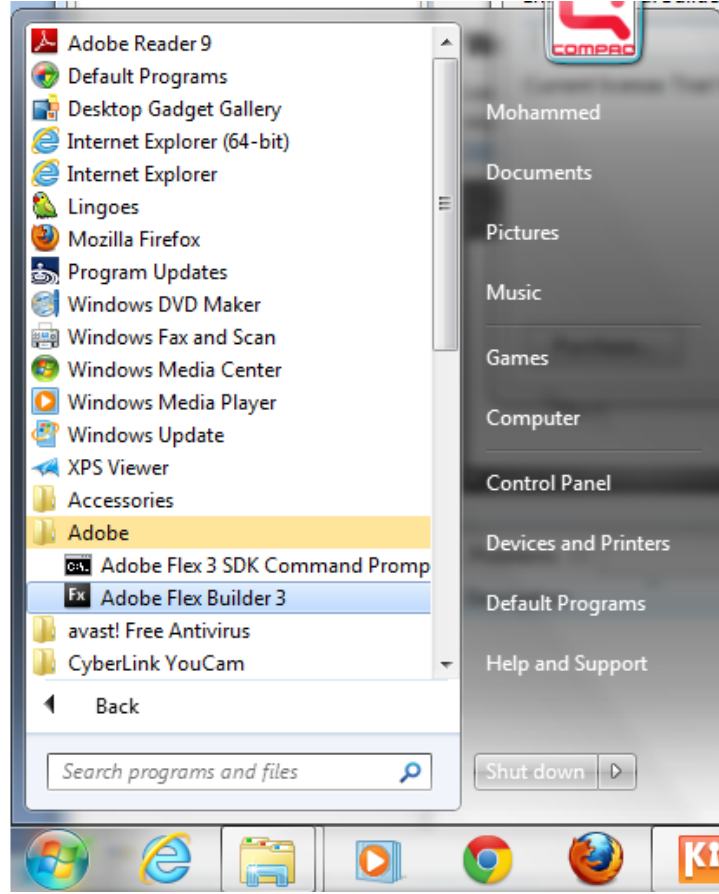
شكل 7-1 نافذة عملية التثبيت

وأخيرا بعد اكمال عملية التثبيت تظهر هذه الشاشة لتوضح لك إذا ما تمت عملية التثبيت بنجاح.



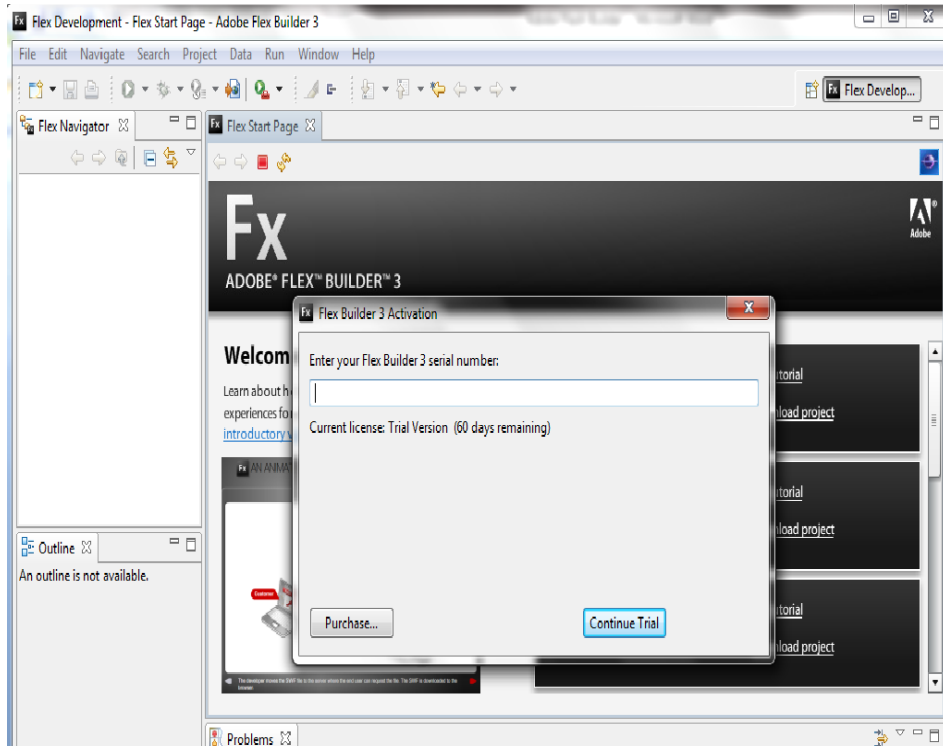
شكل 8-1 نافذة توضح انتهاء عملية التثبيت بصورة ناجحة

بعد ذلك اذهب إلي قائمة إبداء ثم إلي المجلد Adobe واضغط على Adobe Flex Builder 3 لتشغيل البرنامج



شكل 9-1 قائمة إبداء

عند تشغيل البرنامج لأول مرة يطلب منك إدخال السيريال الخاص بالبرنامج وإذا لا تملك سيريال يمكنك استخدام البرنامج بشكل تجريبي لمدة 60 يوم.



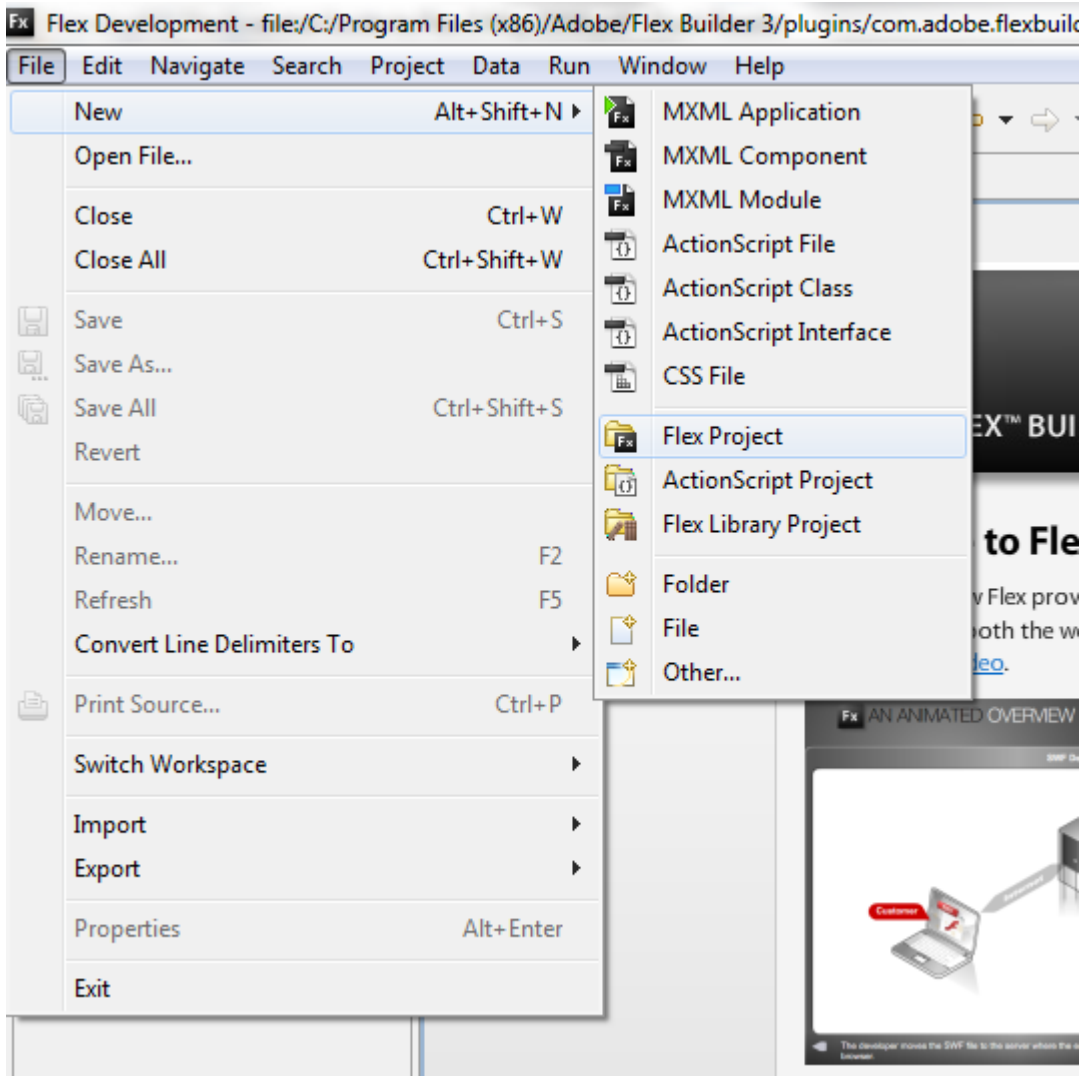
شكل 10-1 النافذة الرئيسية لبرنامج Adobe Flex Builder 3

الفصل الثاني: إنشاء مشروع بلغة الفليكس

Adobe Flash Builder مثلها مثل معظم بيئات التطوير المختلفة تتطلب كتابة البرنامج وجود مشروع محدد قبل تطوير أي تطبيق ، مشاريع الفليكس يتم إدارتها بواسطة بيئة التطوير، حيث تحتوي على كل الملفات اللازمة لبناء واختبار ونشر التطبيقات.

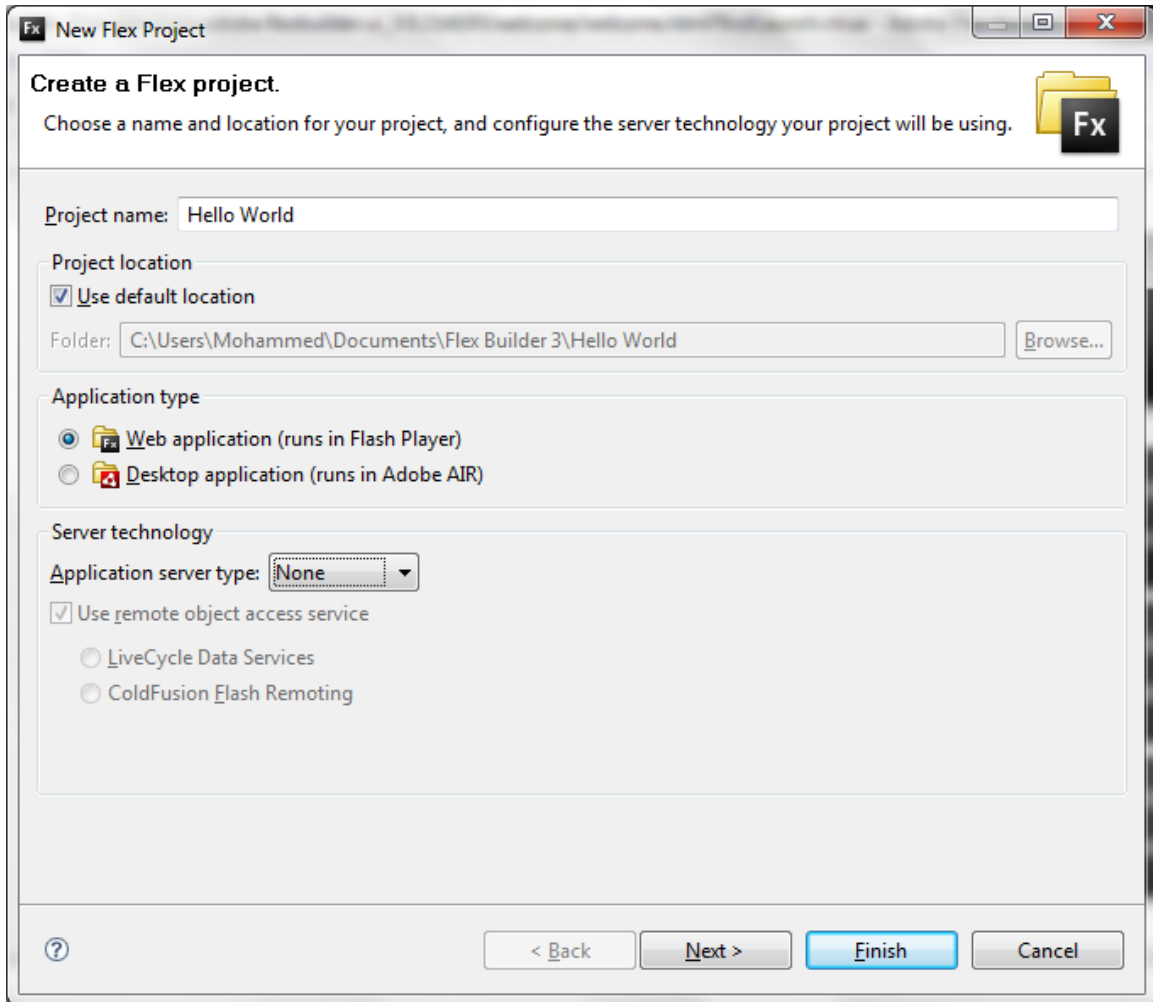
كما في معظم كتب لغات البرمجة، دعنا نبدأ بإنشاء تطبيق بسيط وليكن (Hello World).

1. File > New > Flex Project



شكل 1-2 إنشاء مشروع جديد بواسطة Adobe Flex Builder

2. Create Flex Project



شكل 2-2 صندوق حوار لإنشاء تطبيق فليكس جديد

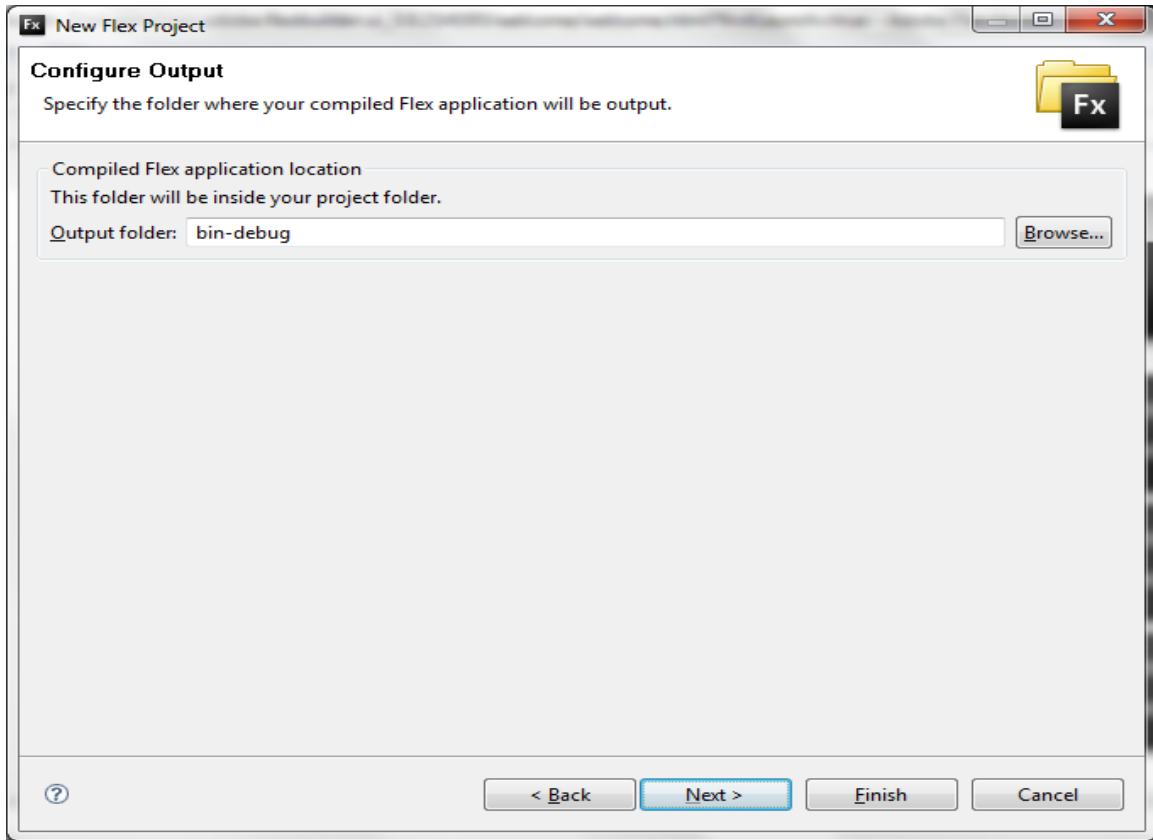
أول حقل خاص باسم المشروع (Project Name)، تحت حقل اسم المشروع حقل موقع المشروع (Project Location) من خلال هذا الحقل يمكنك تحديد المجلد الذي يتم فيه حفظ ملفات المشروع وهذه الملفات ضرورية لتنفيذ المشروع هنا نستخدم الموقع الافتراضي للمشروع.

تحت حقل موقع المشروع حقل نوع التطبيق (Application type)، هناك نوعين من التطبيقات يمكن انشاءها باستخدام Adobe Flex Builder

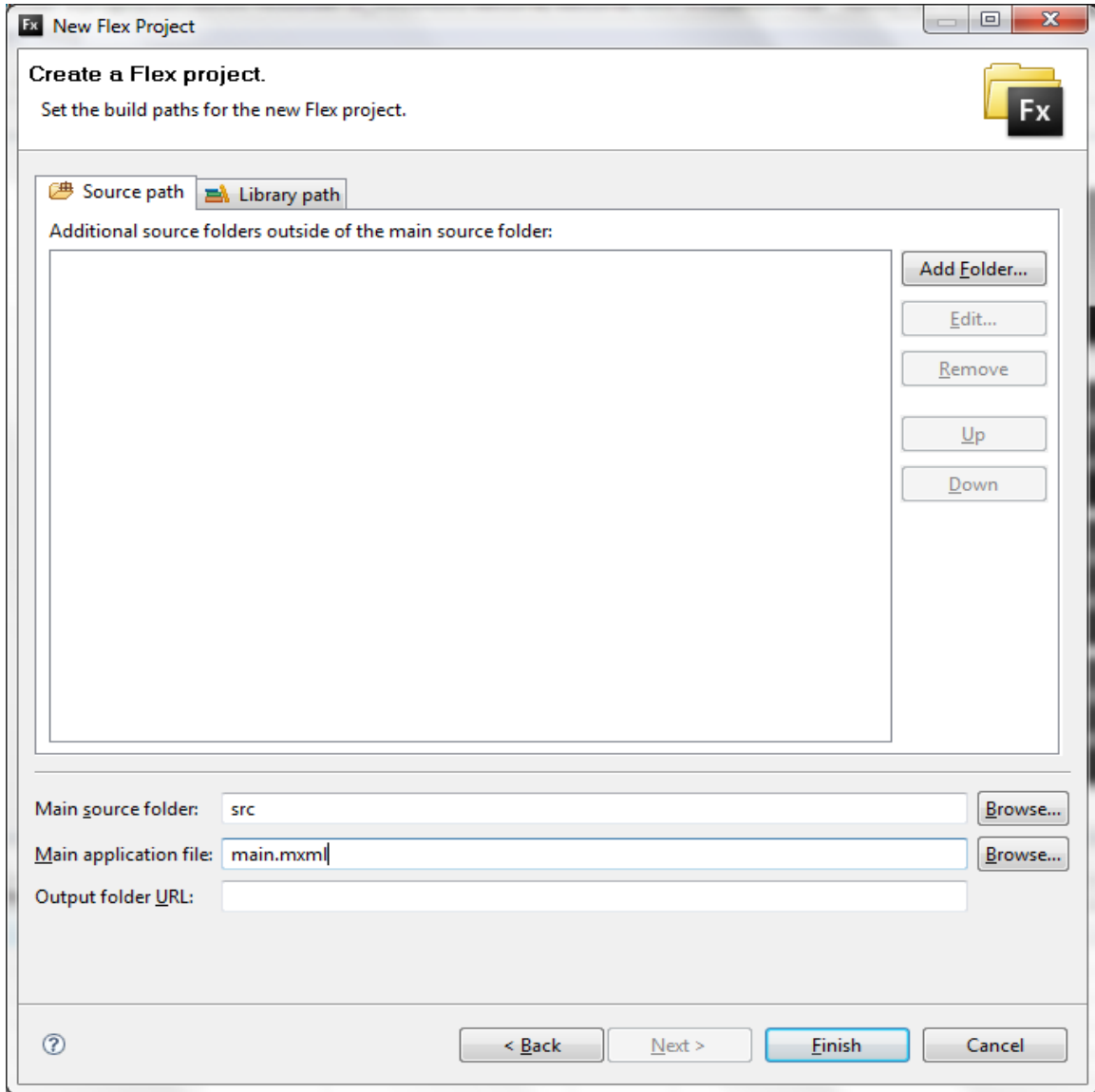
- تطبيقات الانترنت Web application يتم تشغيل هذه التطبيقات بواسطة مشغل الفلاش حيث يتم إنشاء ملفات SWF يمكن عرضها بواسطة متصفح الانترنت.

- تطبيقات سطح المكتب Desktop Application يتم تشغيلها بواسطة Adobe AIR وهي تقنية جديدة أطلقتها شركة Adobe، وتشير كلمة AIR إلى Adobe Integrated Runtime. تدعم الفليكس العديد من لغات الخادم (PHP، JAVA، .NET، ColdFusion)، في هذا التطبيق نترك حقل Application Server type كما هو (None) ونضغط على Next.

3. Configure Output



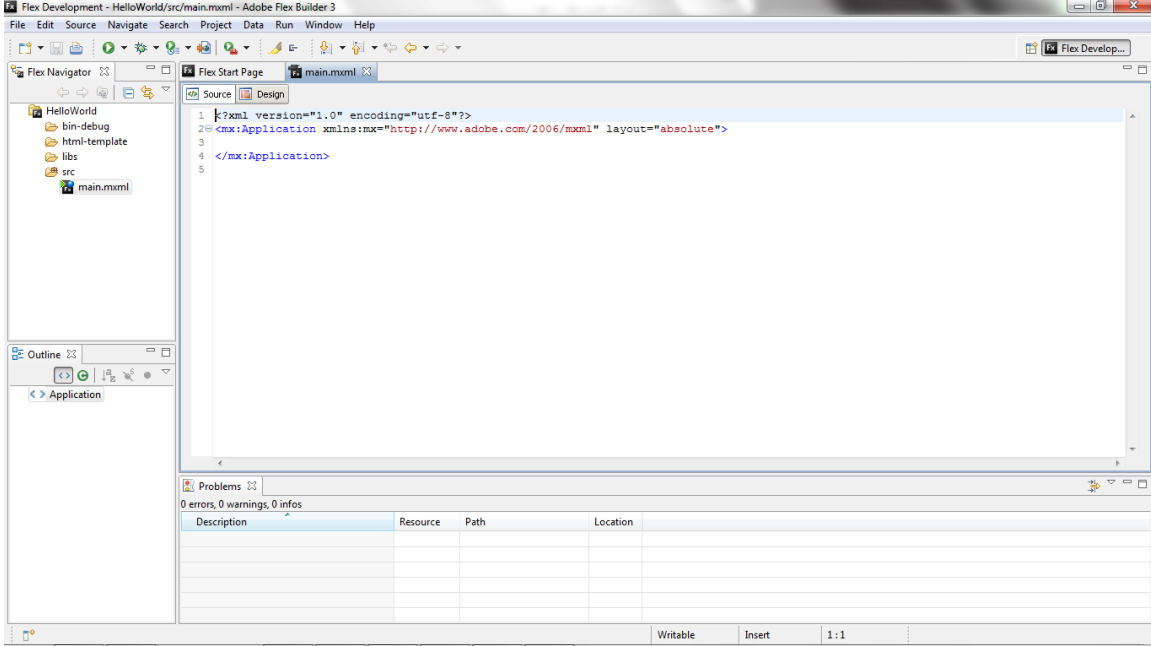
شكل 2-3 المجلد الافتراضي للملفات



شكل 4-2 تسمية ملف التطبيق

كل مشاريع الفليكس تحتوى على ملف واحد بالامتداد MXML يسمى ملف التطبيق Application file وملفات MXML إضافية تسمى مكونات component يتم إنشاء ملف التطبيق بصورة افتراضية وغالبا ما يحمل نفس اسم المشروع.

4. Click Finish



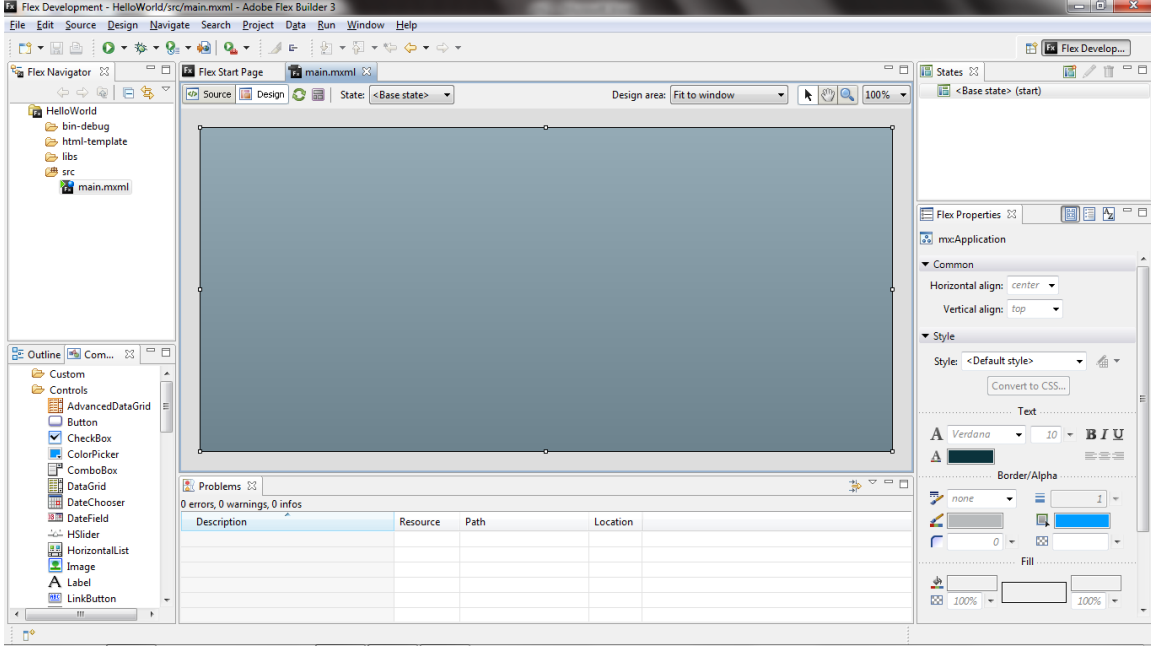
شكل 2-5 واجهة المشروع Flex Builder 3

عندما نقوم بإنشاء مشروع في Adobe Flex Builder 3 يتم فتح واجهة مشروع الفليكس وتتكون من:

Flex Navigator: تحتوى هذه النافذة على الملفات والمجلدات اللازمة لهذا المشروع ، من أهم هذه المجلدات المجلد bin-debug يحتوي هذا المجلد على مخرجات التطبيق، مجلد src يحتوي على ملف التطبيق الرئيسي main.mxml.

Outline: هذه النافذة يتم من خلالها عرض هيكل التطبيق.

Editor View: في هذه النافذة من واجهة تطبيقات الفليكس نقوم بكتابة التعليمات البرمجية الخاصة بالتطبيق ، داخل واجهة الفليكس هناك واجهات فرعية ، عندما ننظر إلي المنصة Editor View نلاحظ القسم الخاص بالتعليمات البرمجية الخاصة بالتطبيق ويسمى نافذة عرض التعليمات البرمجية Source Perspective ولكن إذا نظرنا في الركن العلوي يسارا من هذا القسم نلاحظ ان هناك اثنين من الأزرار Source و Design، الزر Design ينقلنا إلي نافذة عرض التصميم وتسمى Design Perspective.

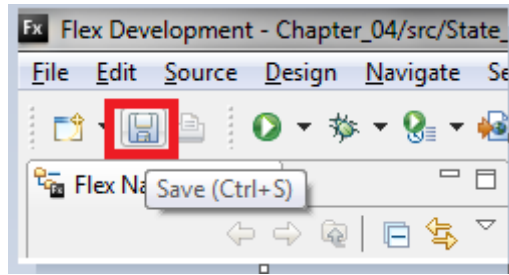


شكل 2-6 نافذة عرض التصميم Design Perspective

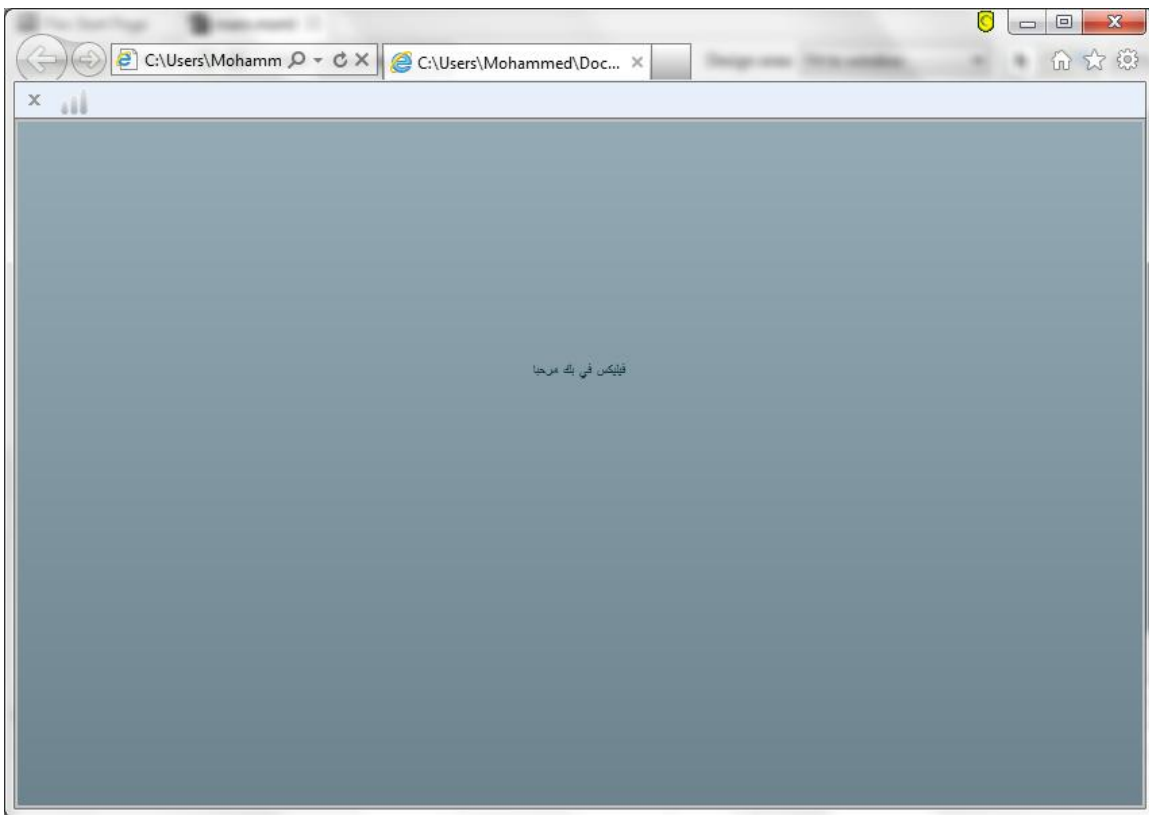
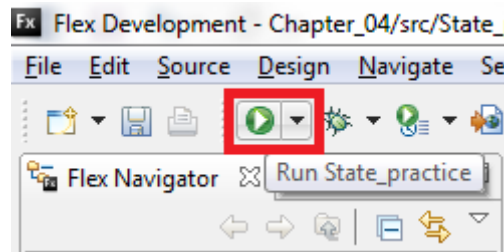
نلاحظ عند فتح نافذة عرض التصميم إن هناك عدد من النوافذ لم تكن موجودة في نافذة عرض التعليمات البرمجية مثل نافذة عرض المكونات Component View، نافذة عرض الحالات State View، نافذة عرض خصائص مكونات فليكس Flex Properties، من خلال نافذة عرض التصميم يمكن إنشاء التطبيقات المرئية كما في Visual Studio، دعنا الآن نقوم بإنشاء تطبيق مرئي بسيط.

نلاحظ ان نافذة عرض المكونات Component View تحتوي على عدد من المجلدات (Controls)، هذه المجلدات تحتوي على المكونات التي نحتاجها لبناء تطبيقات الفليكس، افتح مجلد عناصر التحكم controls يحتوي هذه المجلد على مكونات نحتاجها لإرسال أو استقبال البيانات.

1. من المجلد Controls اختر المكون Label وقم بإدراجه على منصة التصميم Editor View
2. انقر نقر مزدوج Double Click على المكون Label واعد كتابة النص فيه إلى "مرحبا بك في فليكس".
3. يمكنك حفظ المشروع باستخدام الامر Ctrl + S او الضغط على الزر Save من شريط الأدوات.



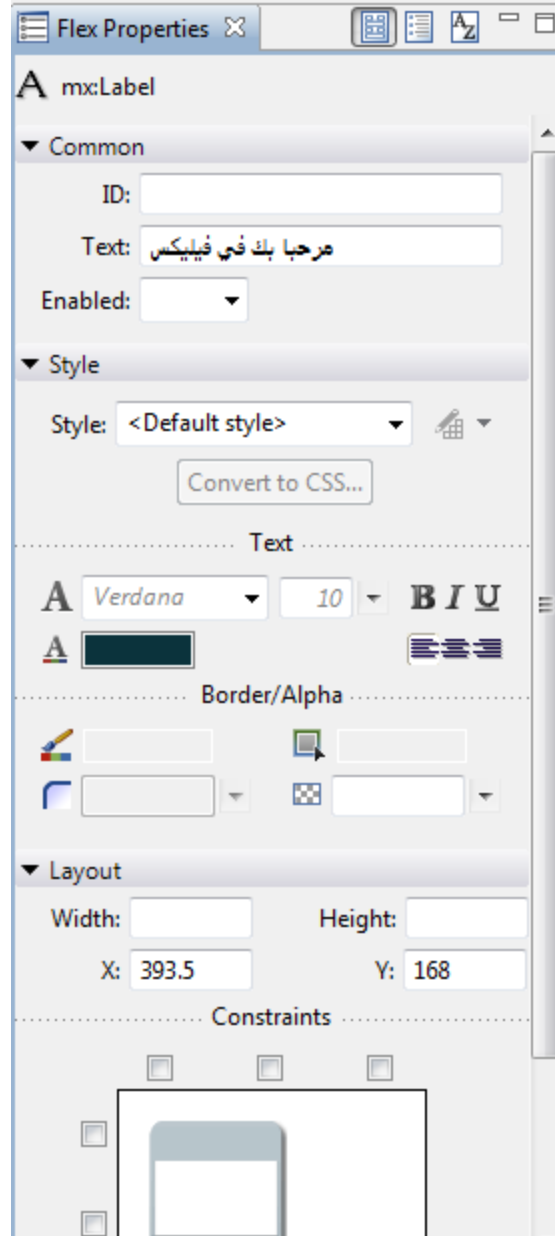
4. لتنفيذ وتشغيل المشروع اضغط على الزر Run من شريط الأدوات



شكل 7-2 ناتج التنفيذ على متصفح الانترنت الافتراضي

تغير خصائص المكونات:

تسمح لك بيئة التطوير فليكس بتغير خصائص المكونات وذلك عن طريق التعليمات البرمجية أو بصور مرئية من نافذة عرض التصميم دعنا نرى كيفية تغير الخصائص بصورة مرئية.



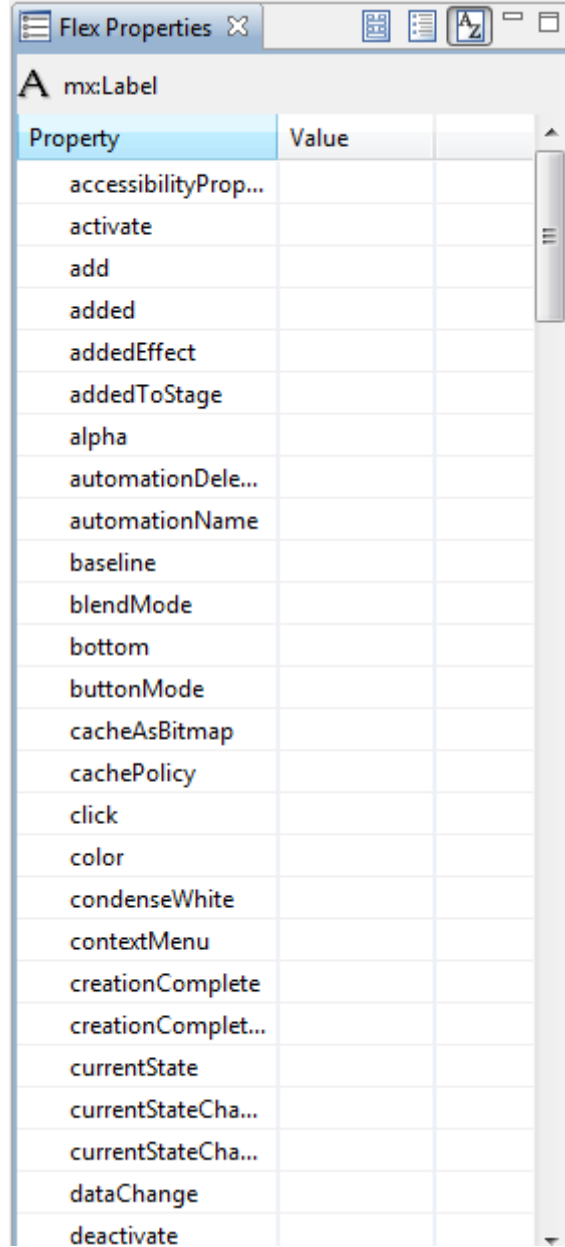
شكل 9-2 نافذة عرض خصائص مكونات فليكس Properties View

تختلف هذه النافذ حسب نوع المكون على سبيل المثال إذا كنا نعمل على زر من النوع Button تكون الخيارات مختلفة، في أعلى هذه النافذة من الجهة اليمنى هناك عدد من الأزرار تستخدم لتحديد طريقة

عرض خصائص المكونات، أول زر هو **Standard View** وهو موضح في الشكل 9-2، الزر الثاني **Category View** في هذه الحالة تكون الخصائص مقسمة في شكل مجموعات، الزر الثالث **alphabetical View** وتكون الخصائص مرتبة أبجديا حسب اسم الخاصية.

Property	Value
Common	
alpha	
condenseWhite	
enabled	
htmlText	
id	
includeInLayout	
selectable	
styleName	
text	... مرحبا بك في
visible	
Effects	
Events	
Layout Constraints	
Size	
Styles	
Other	

شكل 10-2 Category View



شكل 11-2 Alphabetical View

تطبيقات الفليكس :

افتح نافذة عرض التعليمات البرمجية Source Perspective لتشاهد التعليمات البرمجية التي تم توليدها بعد إضافة عنصر التحكم Label.


```
Flex Start Page main.mxml X
Source Design
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
3   <mx:Label x="393.5" y="168" text="مرحبا بك في فيليكس"/>
4
5 </mx:Application>
6
```

شكل 2-12 التعليمات البرمجية في نافذة عرض التعليمات البرمجية

دعنا الآن نقوم بشرح هذه التعليمات، يتم إنشاء تطبيقات الفليكس بواسطة لغة MXML ولغة
ActionScript 3.0.

لغة MXML :

لغة MXML هي لغة معتمدة على مكتبات لغة XML وتلتزم بكل قواعد الأوسمة tags في لغة XML ،
لكنها تضع في الاعتبار البساطة .
من المثال السابق السطر الأول من التعليمات البرمجية:

```
<?xml version="1.0" encoding="utf-8"?>
```

كل ملفات لغة MXML تبدأ Document Type Declaration (DTD) الإعلان عن نوع ملف
xml وهذا يعني ان الملف يتم فحصه للتحقق من تطابقه مع هيكل لغة XML القياسي.

السطر الثاني من التعليمات البرمجية:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"  
layout="absolute">
```

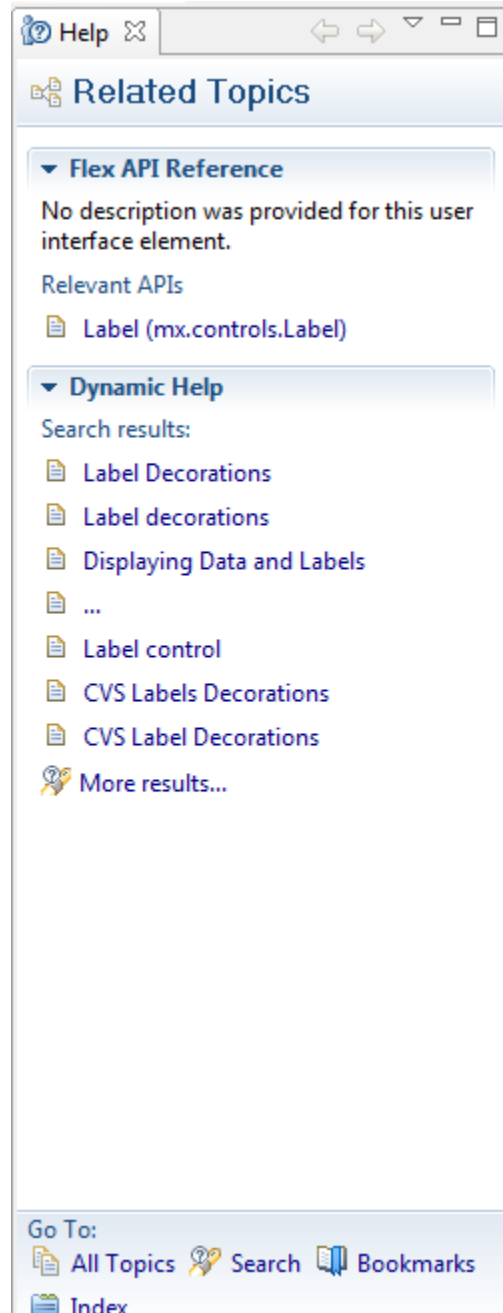
يشير هذا الوسم الي الملف الرئيسي للتطبيق، فقط الملف الرئيسي للتطبيق يجب أن يبدأ بالوسم
application، كل وسم في MXML يقابله ملف فئة class في ActionScript، وكل وسم
يتم فتحه لا بد من إغلاقه كما في لغة xml، وكل وسم يجب ان يتضمن الكلمة mx كما نلاحظ في وسم
عنصر التحكم Label التالي:

```
<mx:Label x="393.5" y="168" text="welcome to flex"/>
```

ولان الوسم الخاص بعنصر التحكم Label لا يحتوي على أي وسم بداخله لذلك تم إغلاقه بشكل
مختزل ./>

دليل المساعدة:

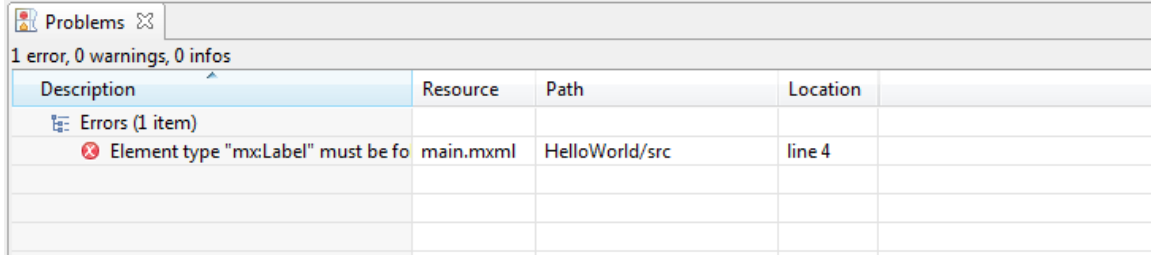
تقدم معظم لغات البرمجة دليل للمساعدة في فهم بنية الفئات التي تمثل اللغة دعنا نرى وثائق المساعدة في الفليكس، للحصول على معلومات عن أي وسم في الفليكس يمكنك وضع مؤشر الفارة على ذلك الوسم والضغط على المفتاح f1 من لوحة المفاتيح.



شكل 13-2 نافذة المساعدة help View

الأخطاء والمشاكل:

أذهب إلى الوسم label وقم بحذف وسم الإغلاق، بافتراض أنك نسيت ذلك، بعد ذلك قم بحفظ التطبيق، بمجرد حفظ التطبيق ستلاحظ ظهور خطأ في نافذة عرض الأخطاء أسفل الواجهة، تقوم الفليكس بفحص التعليمات البرمجية تلقائياً لاكتشاف أي خطأ محتمل.



The screenshot shows the 'Problems' window in an IDE. It displays a table with columns: Description, Resource, Path, and Location. The table contains one error entry: 'Element type "mx:Label" must be fo' in the Description column, 'main.mxml' in the Resource column, 'HelloWorld/src' in the Path column, and 'line 4' in the Location column. The window title is 'Problems' and it shows '1 error, 0 warnings, 0 infos'.

Description	Resource	Path	Location
Errors (1 item)			
✘ Element type "mx:Label" must be fo	main.mxml	HelloWorld/src	line 4

شكل 14-2 نافذة عرض الأخطاء

الفصل الثالث: لغة ActionScript

مع مرور السنين تطورت لغة البرمجة ActionScript من لغة برمجة بسيطة لمعالجة الرسوم والاشكال المتحركة الي شكلها الحالي كلغة برمجة كائنية المنحى OOP قبل ان نبدأ بشرح لغة ActionScript 3.0 نحتاج الي تعلم بعض المفاهيم الاساسية عن البرمجة كائنية التوجه.

مفهوم ال Class:

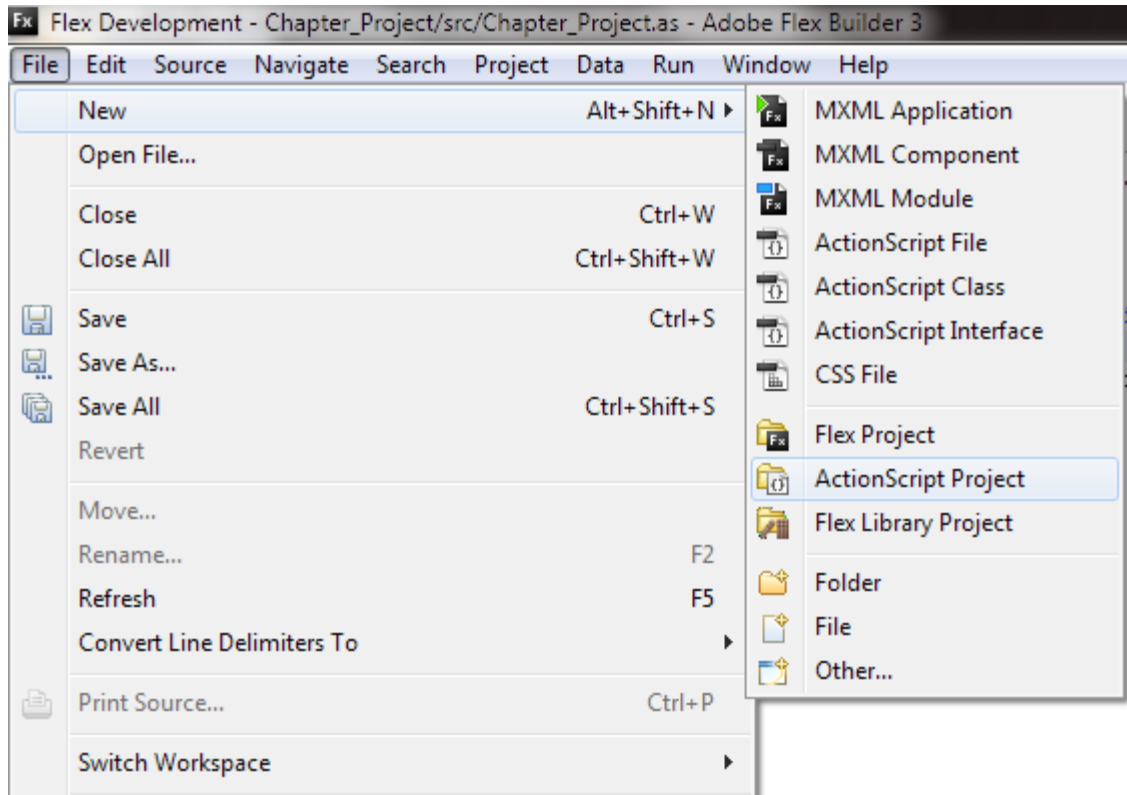
في بداية عهد البرمجة، كان المطورين يعتمدون على تقنية تسمى البرمجة الاجرائية، والمقصود هنا ان كل التعليمات البرمجية اللازمة لبناء المشروع تكون في عدد قليل من الملفات، كل ملف من هذه الملفات يحتوي على الاف وفي بعض الاحيان مئات الالاف من الاسطر من التعليمات البرمجية التي يتم تنفيذها الي حد كبير بشكل متسلسل، ومن لغات البرمجة الاجرائية لغة FORTRAN، Pascal، COBOL، Ada.

في بداية السبعينيات طور دينس ريتش اول لغة برمجة اجرائية في معامل بل تسمى لغة C، بعد حوالي عشرة اعوام من ذلك طور بيارني ستروستراب الجيل التالي من لغة البرمجة C وسميت C++ وهو ايضا باحث في معامل بل، مع هذا الجيل الجديد من البرمجة تم تقديم مفهوم جديد في البرمجة يسمى البرمجة كائنية المنحى OOP، ومن اشهر لغات البرمجة كائنية المنحى لغة Java ، C#

ما يميز البرمجة كائنية المنحى عن البرمجة الاجرائية هو تقسيم العمل، كما ذكرت سابقا ان البرمجة الاجرائية تستخدم سلسلة طويلة من التعليمات البرمجية، بينما في البرمجة كائنية المنحى يتم تقسيم العمل الي وحدات صغيرة تسمى ملف الفئة Class File.

ملف الفئات هو برنامج مستقل قائم بذاته يحتوي على جميع المتغيرات (تسمى الخصائص properties) والدوال اللازمة لأداء مهمة معينة او مجموعة من المهام ذات الصلة، وملف الفئة هو ايضا بمثابة قالب، وعندما نتحدث عن الفئات لا بد لنا الحديث عن الكائنات Objects والكائن يعتبر نسخة مؤقتة من ملف الفئة يتم تخزينه على ذاكرة الحاسوب، وبما ان ملف الفئات برنامج قائم بذاته ويقوم بأداء وظيفة معينة فيمكن استخدامه مرة اخرى في أي مشروع اخر.

قد تكون لاحظت عند انشاء مشروع فليكس جديد في Adobe Flex Builder خيار لإنشاء مشروع ActionScript جديد Creating a new ActionScript Project كما يوضح الشكل التالي:



شكل 1-3 انشاء مشروع ActionScript

ربما قلت لنفسك بما اننا في هذا القسم نتحدث عن ActionScript فمن المنطقي ان ننشئ مشروع ActionScript ولكن سأقول لك لا! او على الاقل ليس في هذه المرحلة.

عند انشاء مشروع ActionScript لا يمكن استخدام التعليمات البرمجية الخاصة بـ MXML بينما يمكن استخدام تعليمات actionscript البرمجية داخل تعليمات MXML، فإثناء مشروع ActionScript جديد قد يحتاجه بعض المبرمجين للتحكم في التعليمات البرمجية، في هذا الكتاب سنعمل على مشروعات الفليكس فقط Flex Project.

عند كتابة تعليمات برمجية بلغة MXML، يتم تحويل هذه التعليمات الي ملف swf وتسمى هذه العملية بالترجمة Compiling واثناء هذه العملية يقوم الفليكس بتحويل تعليمات MXML الي تعليمات ActionScript، لكن كيف يتم ذلك؟

بعد ان تقوم بانشاء مشروع فيليكس جديد ستفتح لك نافذة التعليمات البرمجية الخاصة بـ MXML مع الهيكل القياسي للتطبيق.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Applicationxmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute">

</mx:Application>
```

دعنا الان نرى كيف يتم تحويل ذلك الي تعليمات `ActionScript`.

تقريبا كل وسم من اوسمة لغة `MXML` يقابله ملف فئة في لغة `ActionScript 3.0` مرتبط بهذا الوسم ومعروف ان لكل وسم مجموعة من الخصائص دعنا الان نقوم بإضافة وسم `Label` مع اسناد النص `"!WelcometoFlex"` للخاصية `text` كما موضح في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Applicationxmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute">
    <mx:Label text="Welcome to Flex !" id="mylabel"/>
</mx:Application>
```

لكن في الواقع تقوم الفليكس بكتابة تعليمة `ActionScript` التالية في الخلفية.

```
Var mylabel:Label = new Label();
mylabel.text = "Welcome to Flex !";
```

في المثال السابق الكائن `mylabel` هو نسخة من الفئة `Label` في لغة `ActionScript` والفرق الوحيد يطلق عليه في لغة `MXML` المعرف `"id"`، نلاحظ من خلال المثال السابق نلاحظ ان تعليمات لغة `MXML` البرمجية ابسط من تعليمات لغة `ActionScript 3.0`.

في بيئة التطوير فليكس يمكنك بناء التطبيقات بوحدة من ثلاثة طرق هي:

- استخدام لغة `MXML` فقط.
- استخدام لغة `ActionScript` فقط.
- دمج تعليمات `MXML` مع تعليمات `ActionScript`.

في هذه الكتاب سنركز على الطريقة الثالثة وهي من أكثر الطرق المستخدمة في بناء تطبيقات فليكس لاحظ في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Applicationxmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical">
```

```

    <mx:TextInput id="myName"/>
    <mx:Label text="{myName.text}" id="mylabel"/>
</mx:Application>

```

في هذا المثال قمنا اولا بتغيير الخاصية layout والتي تستخدم لتخطيط الصفحة الي vertical "راسي" في الوسم Application.

وسم MXML:

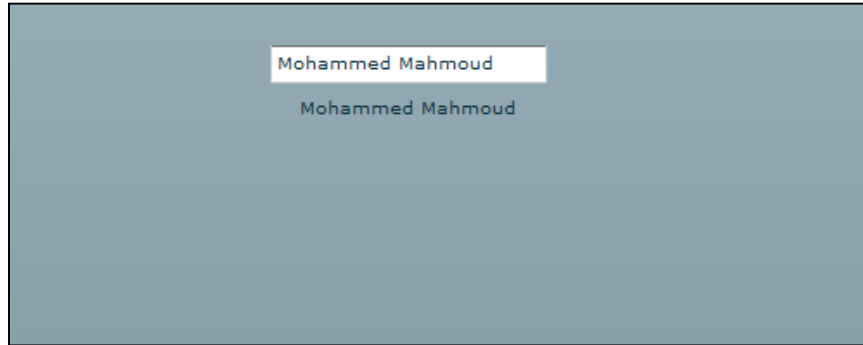
```

<mx:TextInput id="myName"/>

```

قمنا بإنشاء كائن من الفئة TextInput (هي واحدة من عناصر التحكم تستخدم في إدخال النصوص) يأخذ الاسم "myName"، يمكن عرض الوثائق والخصائص المرتبطة بالوسم TextInput بوضع مؤشر الفارة على الوسم والضغط على المفتاح f1 من لوحة المفاتيح كما شرحنا ذلك سابقا مع الوسم Label.

لكن تلاحظ في السطر التالي قمنا بجعل الخاصية text الخاصة بالوسم Label مساوية للخاصية text الخاصة بالوسم TextInput.



شكل 2-3 ناتج التنفيذ

لا تظهر قوة لغة MXML الحقيقية إلا بعد ربطها بلغة ActionScript 3.0 وهذا واضح من المثال السابق، وألان سنوضح كيف يتم دمج لغة MXML مع لغة ActionScript 3.0.

ملاحظة سريعة: يمكن كتابة التعليمات البرمجية الخاصة بلغة ActionScript 3.0 داخل ملف MXML أو كتابتها في ملف منفصل إذا كنت تفضل ذلك.

لكتابة تعليمات ActionScript 3.0 في ملف MXML تحتاج إلي استخدام وسم MXML يسمى Script والذي بدوره يضع وسم يسمى CDATA هذا الوسم يخبر المترجم بمعالجة هذا المحتوى بصورة مختلفة.

تحت الوسم Application قم بكتابة وسم Script،

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="vertical">
  <mx:Script>
    <![CDATA[
      // Action Script Code
    ]]>
  </mx:Script>
  <mx:TextInput id="myName"/>
  <mx:Label text="{myName.text}" id="mylabel"/>
</mx:Application>
```

تتم كتابة التعليمات في لغة Action Script 3.0 البرمجية داخل الوسم CDATA، نقوم أولاً بتعريف المتغيرات، المتغير هو مساحة في الذاكرة يتم حجزها لتخزين البيانات، يبدأ تعريف المتغيرات بالكلمة المفتاحية var، الكلمات المفتاحية هي كلمات محجوزة في لغة البرمجة تستخدم لغرض معين، الجدول التالي يوضح الكلمات المحجوزة في لغة Action Script 3.0.

Break	Final	namespace	Super
Case	Finally	native	Switch
Catch	For	new	This
Class	function	null	Throw
Const	Get	override	To
Continue	If	Package	True
Default	implements	private	Try
Delete	import	protected	Typeof
Do	In	public	Use
Dynamic	Include	Return	Var

Each	instanceof	set	Void
Else	Interface	static	While
Extends	Internal	with	
False	Is		

ملاحظة: بما ان هذه الكلمات محجوزة في لغة ActionScript فلا يمكن استخدامها في تسمية المتغيرات والدوال.

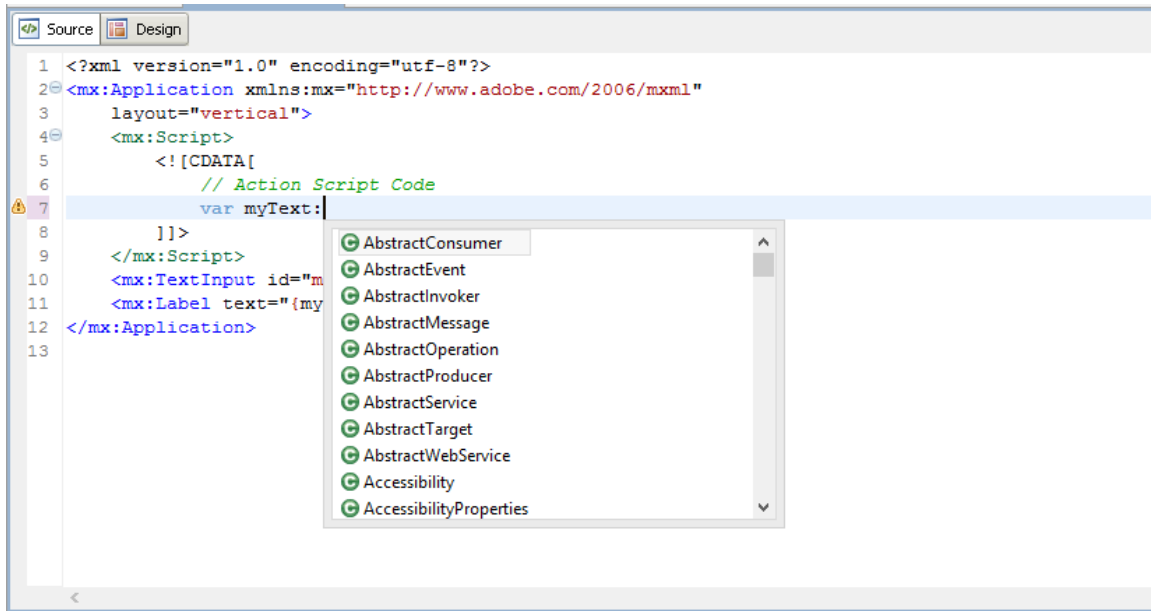
داخل الوسم CDATA قم بتعريف متغير، نقوم بكتابة الكلمة المفتاحية var وبعدها نكتب اسم المتغير هكذا: `var myText`، منطقيا إذا كنا شبهنا المتغير بوعاء يتم فيه تخزين البيانات، وهذا الوعاء صمم لتخزين نوع معين من العناصر، اذن المتغير يجب ان يحتوي على نوع معين من العناصر، لكن كيف نحدد نوع البيانات الخاص بالمتغير؟

في مجال البرمجة هناك ما يعرف بـ `strict typing` وهذا يعنى ان المتغير يتم اسناده الي ملف فئة (نوع بيانات)، يحدد العناصر التي يمكن ان تسند الي المتغير.

من أنواع البيانات الشائعة:

- **Strings:** هي سلسلة من الحروف والعلامات يتم حصرها داخل علامتي تنصيص " "، مثال `"welcome to flex!"`.
- **Numbers:** وهذه فئة واسعة النطاق تشمل الاعداد الصحيحة والاعداد العشرية (كل الاعداد).
- **Boolean:** هذا نوع خاص من أنواع البيانات يأخذ واحدة من قيمتين (`true, false`).

بعد اسم المتغير قم بكتابة نقطتان (:). ليتم عرض قائمة من ملفات الفئات التي يمكن ان تستخدم كأنواع بيانات (انظر الي الشكل التالي:).



شكل 3-3 قائمة ملفات الفئات (أنواع البيانات)

اختر نوع البيانات String من القائمة ليصبح شكل التعليمة البرمجية لتعريف المتغير هكذا `var myText:String` ، وهكذا تم الإعلان عن المتغير لكن من الجيد اسناد قيمة ابتدائية للمتغير، وتسمى هذه العملية تهيئة المتغير (initialization)، يتم اسناد القيمة الافتراضية (0) للمتغيرات الرقمية، وسلسلة فارغة للمتغيرات من النوع String ("") ، و (false) للمتغيرات المنطقية.

`var myText:String = "Mohammed Mahmoud";`

علامة المساو المفردة غالبا تعرف بعامل الاسناد (assignment operator)، وهذا يعني ان كل ما في الجانب الأيمن من علامة المساو يتم اسناده الي الجانب الايسر من علامة المساو.

في البرمجة، التعبير (statement) هو امر الي البرنامج للقيام بشيء ما، من الأفضل في ActionScript ان يتم وضع فاصلة منقوطة (semicolon) في نهاية كل تعبير مع ان ذلك ليس الزاما.

قم الان بربط البيانات، غير قيمة الخاصية text الخاصة بالوسم label الي قيمة المتغير myText، سيكون الشكل النهائي للبرنامج هكذا:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Applicationxmlns:mx="http://www.adobe.com/2006/mxml"
```

```

layout="vertical">
<mx:Script>
    <![CDATA[
        Var myText:String = "Mohammed Mahmoud";
    ]]>
</mx:Script>
<mx:TextInput id="myName"/>
<mx:Label text = "{myText}" id="mylabel"/>
</mx:Application>

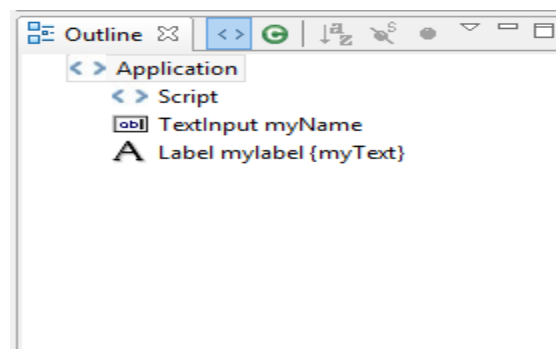
```

قم بحفظ البرنامج وتنفيذه



شكل 4-3 ناتج تنفيذ البرنامج

تحدثنا سابقا عن اهمية نافذة العرض التفصيلي OutLine View عند بناء التطبيقات، فمن الجيد ابقاء العين على المخطط التفصيلي للمشروع اثناء العمل، هذه النافذة موجودة في الركن الاسفل من اليسار في بيئة التطوير Flex Builder، كما تلاحظ في الشكل 5-3 ادناه هذه النافذة تمنحك نموذج مرئي للتسلسل الهرمي لهيكل صفحة MXML كما موضح في الشكل التالي:



شكل 5-3 نافذة العرض التفصيلي

التعليقات Comments:

من الجيد التعليق على التعليمات البرمجية، إذا كنت انت او أي مبرج اخر نظر الي التعليمات البرمجية فمن السهل ان يعرف او يفهم الغرض من كل جزء في التعليمات البرمجية.

تستخدم فليكس التركيب الشائع لجملة التعليق في معظم لغات البرمجة، وهناك نوعين من التعليقات:

- تعليق سطر واحد Single Line Comment:

```
// This is single Line Comment
```

- تعليق أسطر متعددة Multiple Line Comment:

```
/* This is an example of a multiple-line comment,  
   which encompasses several lines in one block */
```

من الجيد كتابة تعليق في بداية الصفحة يشير الي متى ولماذا تم انشاء هذا الملف، كما يمكن تعليق اوسمة لغة XML وبدلا من استخدام الأساليب التي سبق ذكرها يمكن استخدام اسلوب مماثل لتعليق التعليمات البرمجية في لغة XHTML:

```
<!--          -->
```

عادة تستخدم التعليقات لإضافة وصف لتفسير التعليمات البرمجية ومع ذلك لديها استخدام ثاني: تعطيل التعليمات البرمجية الغير مستخدمة مؤقتا وهذا مفيد جدا عند عملية الاختبار، فبدلا من حذف التعليمات البرمجية يمكن تعليقها وعندما نريد استخدامها مرة اخرى فقط يمكنك ازالة التعليق، عند تنفيذ البرنامج يتم تجاهل التعليقات ولا يتم تنفيذها.

الدوال Function:

الدوال هي مجموعة من التعليمات البرمجية التي تقوم بأداء وظيفة محددة ويمكن استدعاها في أي وقت.

تخيل معي انت بحاجة لجمع رقمين و عليك القيام بهذه المهمة عدة مرات في التعليمات البرمجية، ستكون عملية كتابة التعليمات البرمجية مرارا وتكررا امر غير فعال فبدلا من ذلك يمكنك كتابة دالة واحدة ثم تقوم باستدعائها في كل مرة تحتاج الي تنفيذ هذه المهمة، وعلى الرغم من ان هذا الامر يبدو بسيطاً لكن تخيل إذا كنا نحتاج الي تنفيذ هذه المهمة مئات المرات فهذا يعني اننا نحتاج الي كتابة نفس التعليمات البرمجية مئات المرات.

هناك نوعين اساسيين من الدوال: الدوال التي تقوم فقط بأداء وظيفة معينة، والدوال التي تقوم بإرجاع قيمة عندما يتم استدعائها.

اول سطر في تعريف الدالة يسمى التوقيع signature، بعد التوقيع نقوم بفتح واغلاق اقواس متعرجة ({})) والتي تحتوي على التعليمات البرمجية التي تقوم الدالة بتنفيذها ويسمى هذا بجسم الدالة Function Body.

قم بوضع التصريح التالي للدالة تحت تصريح المتغيرات في قسم ال Script:

```
function createMyName():String
{
}
```

سنحدث بالتفصل عن هذا التصريح، اسم الدالة هو createMyName، تقوم هذا الدالة عند استدعاءها بارجاع معلومات من النوع String، تسمى String نوع القيمة الراجعة من الدالة إذا كانت الدالة لا تحتاج الي ارجاع أي شيء، سنقوم بوضع نوع القيمة الراجعة void.

دعنا الان نقوم بوضع بعض التعليمات البرمجية في جسم الدالة createMyName :

```
function createMyName():String
{
    var myFristName:String = "Mohammed";
    var myLastName:String = "Mahmoud";
    return myFristName + " " + myLastName ;
}
```

اول سطرين هما لتهيئة متغيرين من النوع string كما تحدثنا عن هذا من قبل والفرق اننا قمنا بتهيئتها داخل جسم الدالة، من المهم ان تعرف في حالة إنشاء المتغيرات داخل جسم الدالة تسمى بالمتغيرات المحلية Local Variable، وتكون فترة وجود هذه المتغيرات طوال عمل الدالة فقط وتسمى المتغيرات خارج جسم الدالة بالمتغيرات العامة Instance Variable وتكون فترة وجود هذه المتغيرات طوال عمل التطبيق.

في السطر الاخير من جسم الدالة نستخدم الكلمة المفتاحية return، عندما تكون نوع القيمة المرجعة من الدالة غير النوع void، يجب ان تحدد ما سيتم ارجاعه باستخدام الكلمة المفتاحية return، في هذا السطر مثال لعملية ربط السلاسل concatenation وهي تقوم بربط عدد من العناصر مع بعضها، يتم ربط العناصر باستخدام المعامل (+).

فقط تبقى علينا استدعاء الدالة، عملية استدعاء الدالة بسيطة، حيث تتم باستخدام اسم الدالة مع فتح واغلاق اقواس (()) في الأمثلة التالية سنستخدم هذه الاقواس في تمرير الوسائط او العوامل للدالة.

سنقوم بتعديل الوسم Label كما في موضح في المثال التالي:

```
<mx:Label text="{createMyName()}" id="myLabel"/>
```

المثال التالي يوضح التعليمات البرمجية التطبيق بصورة كاملة:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="vertical">
  <mx:Script>
    <![CDATA[
      var myText:String = "Welcome to Flex!";
      function createMyName():String
      {
        var myFristName:String = "Mohammed";
        var myLastName:String = "Mahmoud";
        return myFristName + " " + myLastName ;
      }
    ]]>
  </mx:Script>
  <mx:Label text="{createMyName()}" id="myLabel"/>
</mx:Application>
```

الان قم بتنفيذ البرنامج لمشاهدة القيمة الراجعة في الوسم Label.



شكل 6-3 ناتج تنفيذ الدالة

تمرير الوسائط Passing Parameters:

في كثير من الاحيان نحتاج الي تزويد الدالة بالمعلومات قبل ان تبدأ بالقيام بالمهمة المطلوبة، عند استدعاء الدالة نحتاج الي ارسال هذه المعلومات كوسائط بعد ذلك تقوم الدالة بمعالجتها حسب الحاجة، دعنا الان نأخذ مثال على ذلك.

سنقوم بتغيير الدالة في المثال السابق طبقا للمثال التالي:

```
<mx:Script>
  <![CDATA[
```

```

var myText:String = "Welcome to Flex!";
function createMyName(myFristName:String ,
    myLastName:String):String
{
    return myFristName + " " + myLastName ;
}
]]>
</mx:Script>

```

نلاحظ اننا قمنا بإزالة المتغيرات وقمنا بدلا عنها بوضع عوامل داخل الاقواس يجب تحديد ترتيب العوامل وكذلك نوع بياناتها، في هذه الحالة تضع الدالة الاسم الاول ثم الاسم الخير حسب الترتيب ويجب ان يكون نوع البيانات String، وهذا يتطلب تعديل في استدعاء الدالة. سنقوم بتعديل الوسم Label كما موضح في المثال التالي:

```

<mx:Label text="{createMyName('Mohammed' , 'Mahmoud')}"
id="myLabel"/>

```

الان قم بتنفيذ المشروع من المفترض ان تكون مخرجات التطبيق مشابهة تماما لمخرجات المثال السابق.

سنقوم بإجراء بعض التعديلات على البرنامج السابق قم بحذف الدالة السابقة وبدلا عنها ضع الدالة التالية (هناك خطأ متعمد في المثال التالي):

```

function addNumbers (Number1:Number , Number2:Number):Number
{
    var sum:int = Number1 + Number2 ;
    return "The Sum of the Numbers is : " + sum ;
}

```

قم بتعديل الوسم Label كما في المثال التالي:

```

<mx:Label text="{addNumbers(2 , 3)}" id="myLabel"/>

```

هناك اشياء بسيطة يمكن ملاحظتها خلال هذا المثال بما اننا نقوم بتمرير قيم رقمية فإننا لا نحتاج الي اقواس اقتباس (") اي كانت مفردة او مزدوجة، القيمة الراجعة من الدالة الان من النوع Number. الان قم بحفظ التطبيق ...

اوووووو! هناك مشكلة تمنع ترجمة وتشغل التطبيق كما نشاهد في الشكل التالي:

Description	Resource	Path	Location
1 error, 0 warnings, 0 infos			
Errors (1 item)			
1067: Implicit coercion of a value of	Chapter02....	Chapter02/src	line 9

شكل 7-3 خطأ مكروه

لكن ما الخطاء هنا، تذكر ذلك انه عند استخدام عامل ربط النصوص (concatenation) يتحول نوع القيمة الراجعة مباشرة الي سلسلة String حتى إذا لم تحتوي على اي قيمة من النوع String، لذلك القيمة الراجعة من الدالة في هذه الحالة من النوع String بينما إذا نظرت الي تعريف الدالة فسترى ان نوع القيمة المرجعة المعلن عنها من النوع Number، لذلك نحتاج الي تغيير نوع القيمة الراجعة الي النوع String عند استخدام عامل ربط السلاسل (+).

قم الان بتغيير نوع القيمة المرجعة الي النوع String، بعد تغيير نوع القيمة الراجعة قم بحفظ وتنفيذ التطبيق في هذه الحالة يفترض ان يعمل التطبيق بشكل سليم.

التعامل مع الاحداث Handling Events:

قبل هذه النقطة كانت كل التعليمات البرمجية يتم تنفيذها عند تشغيل التطبيق، لكن في كثير من الاحيان لا نريد تنفيذ بعض التعليمات البرمجية حتى يقع حدث معين.

الحدث (Event) قد يكون الضغط على أحد ازرار الفأرة، الضغط على اي من مفاتيح لوحة المفاتيح، تمرير مؤشر الفأرة على كائن معين، تحميل التطبيق وغيرها من الاحداث، كل الاحداث تتم في جزئين، مستمع الحدث Event listener، معالج الحدث Event handler

مستمع الحدث (Listener) هو شيء ما يقوم بالاستجابة للأحداث، يتنصت الي الاحداث المعينة التي تحدث، عندما يحدث حدث ما، يسمح مستمع الاحداث للدالة بأداء الوظيفة المطلوبة، هذه الدالة تسمى **معالج الاحداث (Handler)**، من أكثر الاحداث الشائعة الحدث Click المرتبط بالكائن Button ويتم عندما يضغط المستخدم على زر الفأرة، دعنا نقوم بإجراء بعض التعديلات على المثال السابق لنرى ذلك.

قم بإضافة الوسم Button تحت الوسم Label كما في المثال التالي:

```
<mx:Label text="{addNumbers(2 , 3)}" id="myLabel"/>
<mx:Button label="Add the Numbers"/>
```

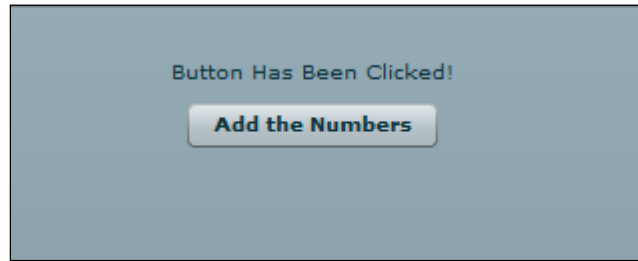

تستخدم الخاصية label في الوسم Button لوضع نص على عنصر التحكم Button، سنتحدث لاحقا عن معالجة الاحداث بصورة أكثر تفصيلا، قدمت MXML طريقة سهلة لمعالجة الاحداث باستخدام الـ inline handler وهذا يعني وضع معالج الاحداث داخل وسم MXML.

كشرح بسيط نحن نريد تغيير الخاصية text في الوسم Label عند النقر على الزر Button قم بإضافة التعليمة البرمجية التالية الي الوسم Button:

```
<mx:Button label="Add the Numbers" click="myLabel.text = 'Button Has Been Clicked!'" />
```

نلاحظ انه عندما نريد وضع سلسلة string داخل string نستخدم اقواس الاقتباس المفردة في السلسلة الداخلية.

قم الان بتنفيذ البرنامج، وقم بالضغط على الزر Button



شكل 3-8 الخرج عند الضغط على عنصر التحكم Button

من الجيد جمع مستمع ومعالج الحدث داخل وسم MXML للمهام البسيطة كما رأينا في المثال السابق، لكن إذا كنا نريد إنجاز مهام أكثر تعقيدا في هذه الحالة نحتاج الي كتابة معالج الاحداث بصورة مستقلة. معالج الاحداث في الحقيقة هو دالة تقوم بالاستجابة للأحداث، هناك عدد من الطرق المختلفة لكتابة معالج الحدث.

الان سنقوم بعمل مثال بسيط يوضح ذلك.

قم بتعديل الدالة addNumbers() في المثال السابق كما موضح في المثال التالي:

```
<mx:Script>
    <![CDATA[
        var myText:String = "Welcome to Flex!";
        function addNumbers (Number1:Number ,
                               Number2:Number) :void
        {
            var sum:int = Number1 + Number2 ;
            myText = "The Sum of the Numbers is : " + sum ;
        }
    ]]>
```

```
</mx:Script>
```

سنقوم بعمل بعض التعديلات على اوسمة MXML، نحتاج الي ربط المتغير myText بالوسم Label، ونحتاج الي استدعاء الدالة addNumbers() عند الضغط على عنصر التحكم Button.

```
<mx:Label text="{myText}" id="myLabel"/>
<mx:Button label="Add the Numbers" click="addNumbers(2 , 3)"/>
```

نتوقع عند بداية تنفيذ هذا البرنامج يتم وضع القيمة

الابتدائية للمتغير myText، بعد ذلك يتم جعل قيمة الخاصية text لعنصر التحكم Label مساوية لقيمة المتغير myText، عند الضغط على عنصر التحكم Button يتم استدعاء الدالة addNumbers() والتي تقوم بجمع الرقمين 2 و 3 ويتم اسناد نتيجة الجمع الي المتغير myText وبالتالي تتغير الخاصية text للكائن Label.

الان قم بتنفيذ البرنامج وإضغط على الكائن Button، اووووه! لماذا لم يحدث اي تغير؟ ،،، الإجابة ستكون من خلال الموضوع التالي:

استخدام الوسم [Bindable]:

عند تشغيل التطبيق السابق، المتغير myText ينقل القيمة الاصلية المخزنة فيه الي كل عناصر التحكم التي تحتاج الي هذه القيمة (في هذه الحالة المتغير myLabel) ومع ذلك فان المتغير لا ينقل التعديلات التي تتم على قيمته تلقائيا للكائنات الأخرى، لكن إذا كنا نرغب في القيام بذلك نحتاج الي استخدام وسم البيانات الوصفي ([Bindable] (ActionScript 3.0 metadata tag).

ببساطة، وسم البيانات الوصفي هو تعليمة برمجية في لغة ActionScript 3.0، عندما يشاهد مترجم لغة ActionScript 3.0 وسم البيانات الوصفي تلقائيا يقوم بكتابة التعليمات البرمجية الضرورية في الخلفية وبالتالي عندما يتم تغير قيمة المتغير هذه التغيرات تلقائيا يتم نشرها الي اي عنصر يستخدم قيمة هذا المتغير.

يجب وضع وسم البيانات الوصفي [Bindable] قبل التعريف لكل متغير يحتاج الي نشر قيمته تلقائيا. الان قم باضافة وسم البيانات الوصفي [Bindable] داخل ActionScript 3.0 كما في المثال التالي:

```
<mx:Script>
  <![CDATA[
    [Bindable]
    var myText:String = "Welcome to Flex!";
    function addNumbers(Number1:Number , Number2:Number):void
    {
      var sum:int = Number1 + Number2 ;
      myText = "The Sum of the Numbers is : " + sum ;
    }
  ]]>
```

```

    }
  ]]>
</mx:Script>

```

الآن قم بتشغيل التطبيق، يجب أن تتغير قيمة الكائن Label وفقا لنتائج الدالة addNumbers عند الضغط على الكائن Button.

التفاعل مع المستخدم:

من المثال السابق، دعنا نسمح للمستخدم بإدخال قيم الرقمين المراد جمعهم نقوم بإضافة عنصر تحكم تحدثنا عنه بشكل موجز سابقا هو عنصر التحكم TextInput.

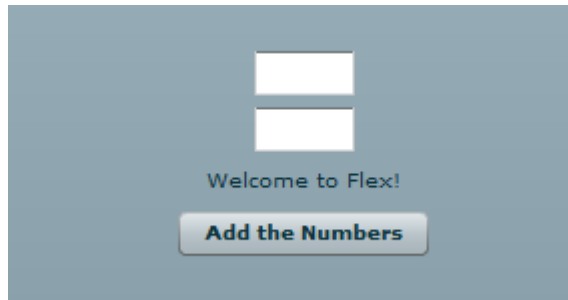
1. قبل الوسم Label قم بإضافة اثنين وسم من النوع TextInput وبما أننا نحتاج الي التفاعل مع هذه العناصر بواسطة ActionScript فيجب ان نعطي كل وسم الخاصية Id في هذا المثال InputNum1 و InputNum2، كما نجعل الخاصية width تساوي 50 Pixel.

```

<mx:TextInput id="InputNum1" width="50"/>
<mx:TextInput id="InputNum2" width="50"/>
<mx:Label text="{myText}" id="myLabel"/>
<mx:Button label="Add the Numbers" click="addNumbers(2,3)"/>

```

2. قم بتشغيل التطبيق الآن ويجب ان يكون الناتج كما في الشكل التالي:



شكل 9-3 شكل النموذج Form

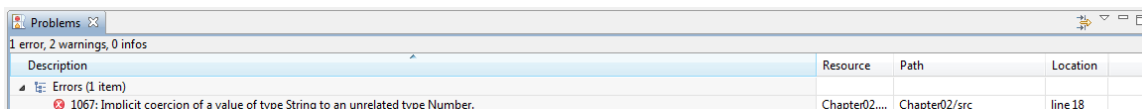
3. الآن سنقوم بتعديل الحدث click للوسم Button كما في المثال التالي:

```

<mx:Button label="Add the Numbers"
click="addNumbers(InputNum1.text, InputNum2.text)"/>

```

4. الآن قم بحفظ التطبيق، لا بد أنك لاحظت علامة الخطأ الحمراء بجانب هذا السطر الموضحة في الشكل التالي:



شكل 10-3 نافذة عرض الأخطاء توضح انه لا يمكن معالجة القيم النصية

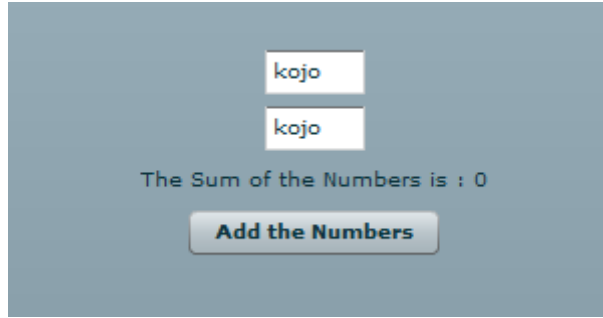
الخاصية text الخاصة بعنصر التحكم InputText تحمل نوع البيانات string، نحن قمنا بتمرير قيمتين من النوع String الي الحدث click الخاص بعنصر التحكم Button وبعد ذلك مررنا هذه القيم النصية الي الدالة addNumbers لكن وسائط هذه الدالة معرفة من النوع Number لذلك نحتاج الي تحويل هذه القيم من النوع String الي النوع Number ويمكن عمل ذلك بالقيام ببعض التعديلات البسيطة .

5. قم بتعديل التعليمة البرمجية كما في المثال التالي:

```
<mx:Button label="Add the Numbers"
click="addNumbers (Number (InputNum1.text) , Number (InputNum2.text)) " />
```

6. قم بتشغيل التطبيق الآن، يجب ان يعمل التطبيق بشكل سليم

قم بإدخال قيم نصية واضغط على عنصر التحكم Button من المفترض ان يحدث Crashing لكن الناتج يكون 0 خلافا للغات البرمجة الاخرى، لان فليكس لها طريقة رائعة لمعالجة الاحداث غير المتوقعة والتي تسمى الاستثناءات Exceptions كما موضح في الشكل التالي:



شكل 3-11 ادخال قيم نصية

محددات الوصول Access Modifiers:

يجب ان يكون لكل كتغير او دالة محدد وصول هذا المحدد يحدد من يمكن ان يصل او لا يمكن ان يصل الي المتغير، إذا كنت جديد في عالم البرمجة فهذا لا يعني لك الكثير، لكن ستفكر ما هي البيانات التي لا يمكن الوصول اليها من قبل الكائنات الأخرى وما هي البيانات التي يمكن الوصول اليها من بقية الكائنات.

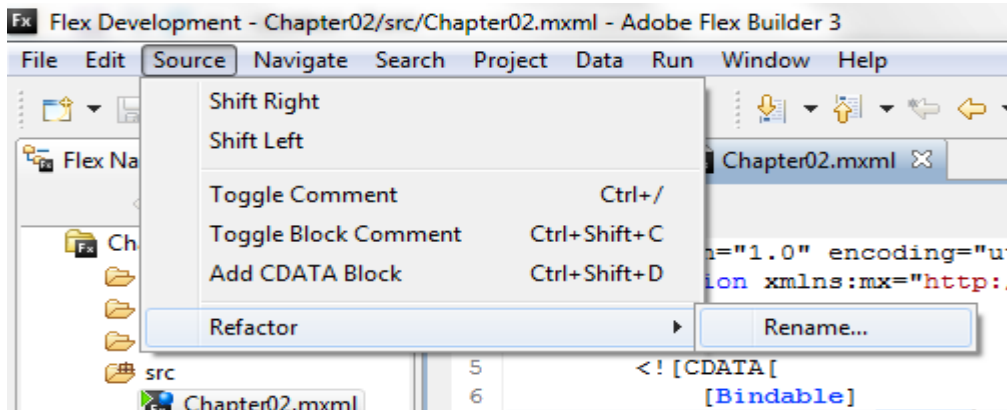
من اكثر محددات الوصول المعروفة محدد الوصول Private ومحدد الوصول Public، محدد الوصول Private يعني انه فقط المتغيرات والدوال داخل نفس ملف MXML يمكن ان تصل الي المتغير.

محدد الوصول Public تعني انه يمكن الوصول الي المتغير من اي ملف اخر داخل المشروع ، في هذه الامثلة عدم استخدام محدد الوصول لا يسبب مشكلة لان المتغيرات والدوال لا يتم الوصول اليها من ملفات اخرى لاحقا سنشرح محددات الوصول بصورة اكثر تفصيلا .
سنقوم بتعديل محدد الوصول في المثال السابق للدالة addNumbers والمتغير myText كما في المثال التالي:

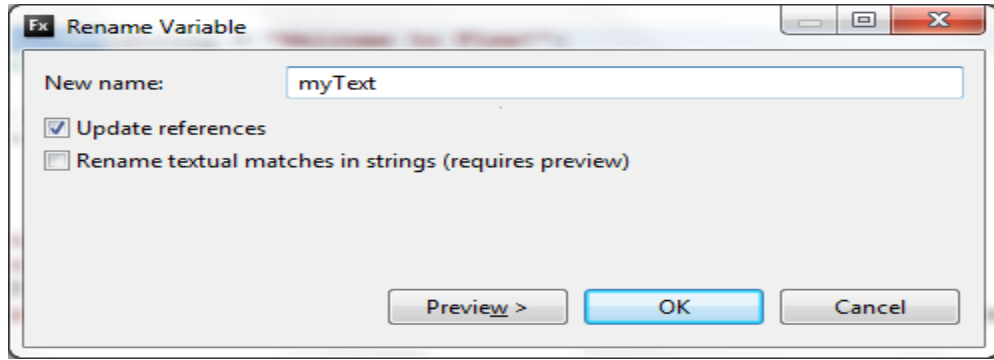
```
<mx:Script>
  <![CDATA[
    [Bindable]
    public var myText:String = "Welcome to Flex!";
    public function addNumbers(Number1:Number , Number2:Number):void
    {
      var sum:int = Number1 + Number2 ;
      myText = "The Sum of the Numbers is : " + sum ;
    }
  ]]>
</mx:Script>
```

إعادة الهيكلة Refactoring:

في المشروعات الكبيرة في بعض الاحيان تحتاج الي تغيير اسم المتغير وهذا المتغير تم استخدامه عدة مرات، هذا قد يسبب مشاكل كثير في المشروع ويحتاج اعادة تصحيح.
معظم بيئات التطوير المتكاملة الحديثة توفر أداة فعالة لمعالجة ما يعرف بإعادة الهيكلة، وهذا ببساطة يعني انه إذا قمنا بتغيير اسم المتغير في مكان ما داخل المشروع، يمكن من خلال بيئة التطوير المتكاملة IDE تغييره في اي مكان اخر، Adobe Flex Builder 3 تحتوي على أداة فعالة لإعادة الهيكلة، دعنا الان نقوم بتجريب ذلك بتغيير اسم المتغير myText الي myNameText.
1. في تعليمات ActionScript قم بتظليل المتغير myText.
2. اختر من القائمة Source < Refactor < Rename كما موضح في الشكل التالي:



شكل 12-3 اختيار أداة إعادة الهيكلة



شكل 13-3 صندوق حوار إعادة تسمية المتغير

نلاحظ ان المتغير الذي تم تحديده تم ادراجه تلقائيا في حقل اسم المتغير الجديد New name.

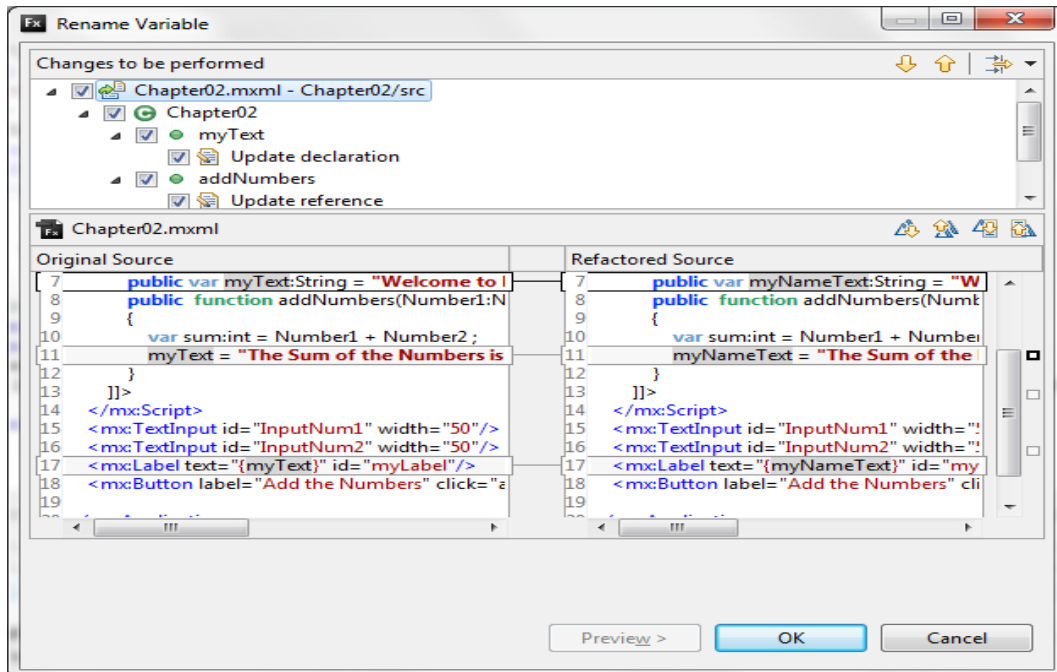
3. سنقوم بتغيير اسم المتغير من myText الي myNameText

ليس ضروريا، ولكن يمكن عمل مراجعة للتعديلات وذلك من خلال الضغط على الزر Preview

4. اضغط على الزر Preview، سيظهر لك صندوق الحوار الموضح في الشكل 14-3، ستشاهد

التعليمات البرمجية قبل التعديل في الجانب الايسر والتعليمات البرمجية بعد التعديل في الجانب

الايمن من صندوق الحوار.



شكل 14-3 صندوق حوار مراجعة المتغيرات

الفصل الرابع: وعاء (حاوية) التطبيق (Application Container)

في عام 1995 قدمت شركة Sun Microsystem مجموعة من الفئات إلي بيئة تطوير الجافا تحت الحزمة Swing، تسمح هذه الفئات للمبرمج بتصميم واجهات رسومية للمستخدم، من أجل كتابة وقراءة البيانات من التطبيقات، والفائدة من وراء هذه المجموعة من الفئات البسيطة واضحة بشكل ملحوظ ، يمكن للمطورين استخدام هذه الفئات لإنشاء الحاويات أو الصناديق من مختلف الأنواع، لكل حاوية دالة تسمى مدير التخطيط (Layout Manager) والتي تقوم بتنظيم محتويات الحاوية بشكل تلقائي ، وهذه المحتويات يمكن ان تكون اي من حقول نماذج إدخال البيانات (, Label , TextInput , Button).

تستخدم الفليكس نفس المفهوم بأشكال مختلفة لتنسيق الحاويات، سنرى كيف يتم ذلك، لكن قبل ان نبدأ يجب ان نقوم ببعض الخطوات المألوفة.

انشئ مشروع جديد تحت مسمى chapter4_project، مع حفظ ملفات المشروع في الموقع الافتراضي والاسم الافتراضي لملف التطبيق MXML.

مدير التنسيق (Layout Manager):

كل الحاويات لها دالة جاهزة تسمى مدير التنسيق، حيث تحدد كيف يتم وضع محتويات الحاوية، هنالك ثلاثة طرق.

- **Absolute**: عند استخدام هذا التنسيق يجب ان يتم تحديد موقع المحتوى بواسطة الاحداثيات x و y.
- **Vertical**: عند استخدام هذا التنسيق يتم وضع المحتويات راسيا وفي وسط الحاوية.
- **Horizontal**: عند استخدام هذا التنسيق يتم تنظيم المحتوى افقيا من اليسار الي اليمين.

دعنا نقوم بتجربة ذلك في مثال بسيط لنرى كيف يتم تنسيق او تخطيط التطبيقات، اذا نظرنا الي

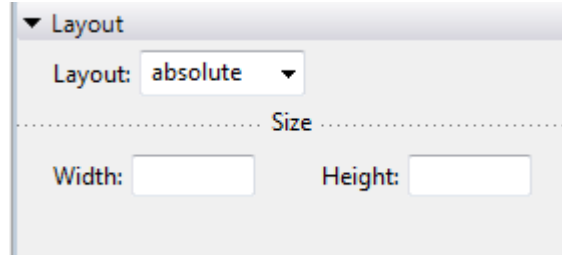
الخاصية layout في الوسم application نجد انها تساوي absolute

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
```

1. انتقل الي نافذة عرض التصميم (Design Perspective)

2. انقر على أي مكان في منصة التصميم

إذا نظرنا الي نافذة عرض خصائص مكونات فليكس Flex Properties سنلاحظ الحقل Layout الخاص بتنسيق الصفحة في القسم الخصائص الخاص بالتنسيق Layout من نافذة عرض خصائص فليكس Flex Properties كما موضح في الشكل التالي:



الشكل 1-4 نافذة تنسيق الصفحة

3. إذا لزم الام قم بتغيير تنسيق الصفحة الي absolute، سيؤدي ذلك الي تعديل الخاصية Layout تلقائيا في الومس Application.

4. اذهب الي القسم الخاص بعناصر التحكم Control من نافذة عرض المكونات Components، وقم بأدراج ثلاثة عناصر تحكم من النوع Label في منصة التصميم كما موضح في الشكل التالي:



الشكل 2-4 ادراج ثلاثة عناصر تحكم من النوع Label

عندما نستخدم التنسيق absolute نحتاج الي تحديد الاحداثيات x و y، تقوم Flex Builder بمعالجة ذلك تلقائيا عند وضع العناصر على منصة التصميم.

5. انتقل الي نافذة عرض التعليمات البرمجية Source.


```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:Label x="252" y="50" text="this is first Label"/>
  <mx:Label x="167" y="135" text="this is second Label"/>
  <mx:Label x="360.5" y="250" text="this is third Label"/>
</mx:Application>
```

ستلاحظ الخاصية x و y للوسم Label

6. للتحقق من النقاط x و y ، قم بحذف الخصائص x و y من عناصر التحكم Label الثلاثة، بعد ذلك قم بتشغيل التطبيق او انتقل الي نافذة عرض التصميم ، ستكون النتيجة كما موضح في الشكل ادناه:



الشكل 3-4 التنسيق Absolute بدون الخاصية x و y

عندما لا نحدد الخاصية x و y في وضع التخطيط absolute افتراضيا تكون 0,0 ، سنقوم الان بتجربة طرق التخطيط الاخرى

7. اذهب الي نافذة عرض التعليمات البرمجية، وقم بتغيير الخاصية Layout في الوسم

Horizontal الي Application.

8. قم بإعادة تنفيذ التطبيق، من المفترض ان يكون الناتج كما في الشكل التالي:

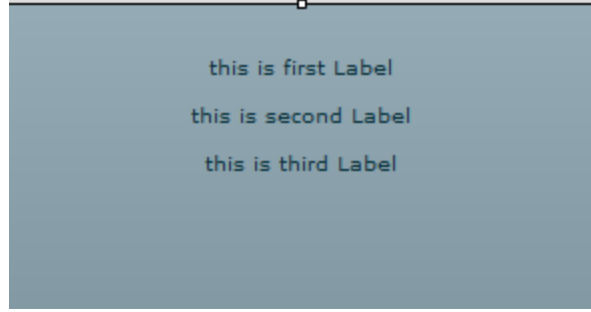


الشكل 4-4 التنسيق Horizontal

ستلاحظ انه تم ترتيب عناصر التحكم Labels بشكل افقي من اليسار الي اليمين

9. اذهب الي نافذة عرض التعليمات البرمجية، وقم بتغيير الخاصية layout الي Vertical

10. قم بتنفيذ التطبيق، من المفترض ان يكون الناتج كما موضح في الشكل التالي:



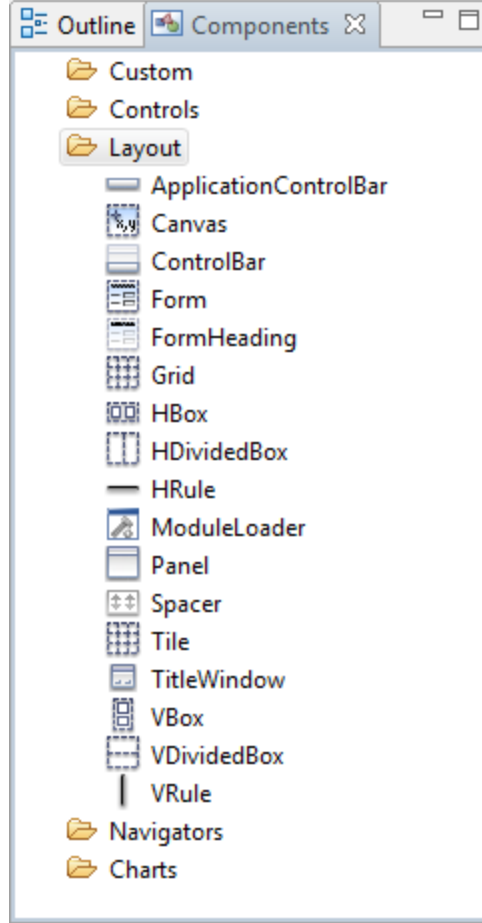
الشكل 4-5 التنسيق Vertical

حاويات التنسيق Layout Containers:

حاوية التنسيق هي رسم مساحة مستطيلة تحدد حجم وموقع الحاويات التابعة او عناصر التحكم التابعة للحاوية .

1. اذهب الي نافذة عرض التصميم ، وقم بحذف عناصر التحكم Label وذلك بتحديد العنصر والضغط على المفتاح delete من لوحة المفاتيح .

2. من نافذة عرض المكونات Component View ، اختر القسم Layout، نلاحظ مجموعة من حاويات التنسيق تحت هذا القسم ، كما موضح في الشكل التالي:



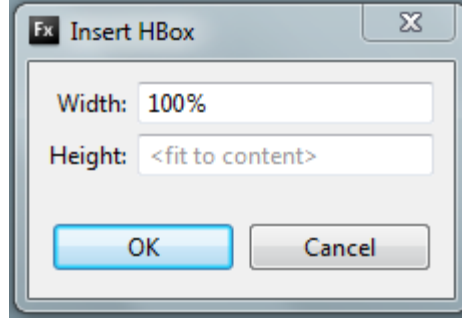
شكل 4-6 حاويات التنسيق

سنركز في هذا الفصل بصورة اكبر على اكثر هذه الحاويات استخداما HBox و VBox

HBox و VBox:

في الحاويات HBox و VBox تكون ادارة تنسيق الحاوية محددة مسبقا ولا يمكن تغييرها، ربما يكون التنسيق واضح من التسمية ، لكن بالنسبة HBox يكون مدير التنسيق محدد بشكل مسبق Horizontal ، بينما مدير التنسيق للـ VBox يكون محدد بشكل مسبق Vertical ، دعنا نقوم بتجربة ذلك:

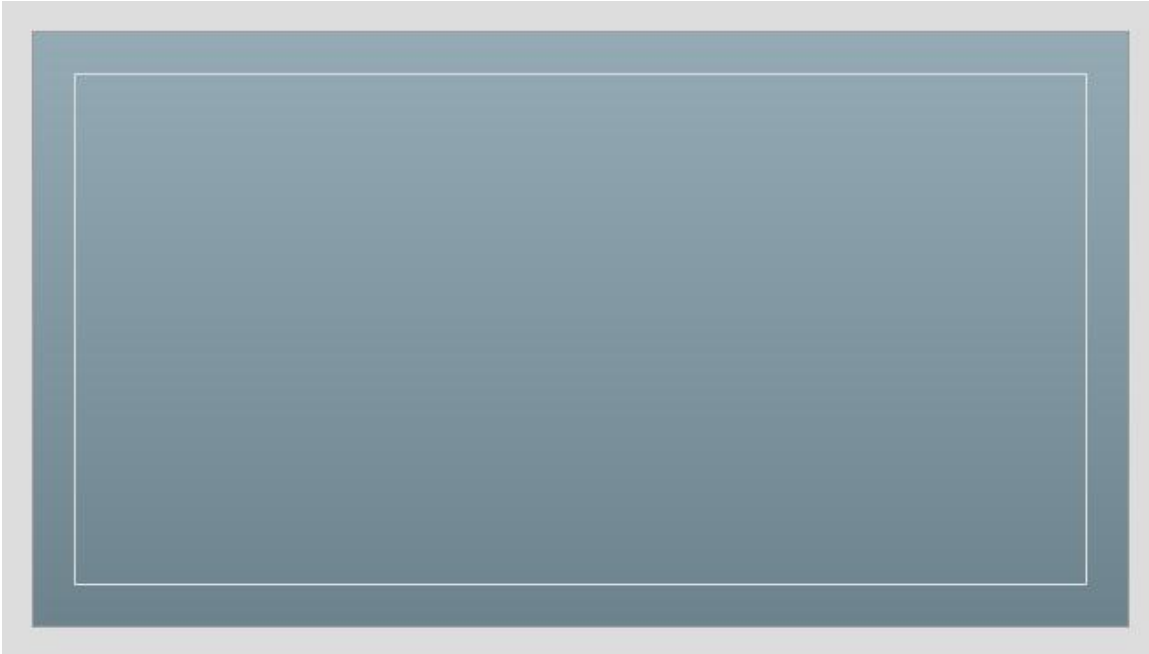
1. اذهب الي نافذة عرض التصميم.
2. قم بتغيير تنسيق حاوية التطبيق الي Horizontal .
3. من نافذة عرض المكونات Component View ، قم بإدراج العنصر HBox. ستلاحظ ظهور صندوق الحوار التالي:



الشكل 4-7 صندوق حوار ادراج العنصر HBox

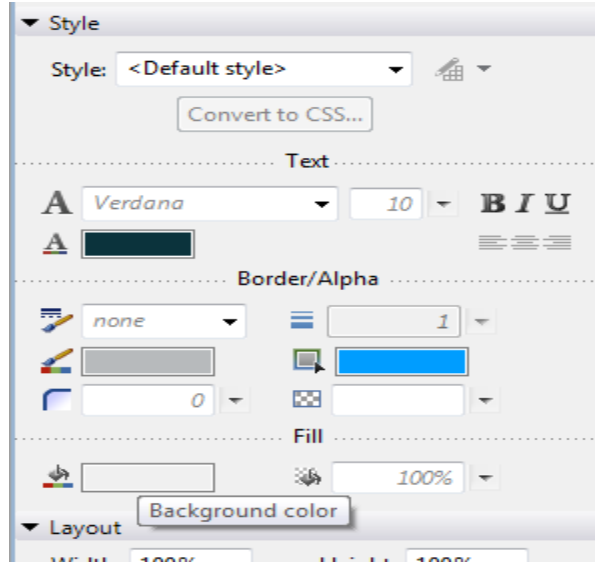
يمكن ضبط حجم الحاوية باستخدام وحدة القياس Pixel او النسبة المئوية من حجم الحاوية الام ، اي بمعنى الحاوية الخارجية لهذه الحاوية

4. ضع كل من الارتفاع Height والعرض Width بنسبة 100% ، واضغط على Ok ، سيكون شكل الحاوية كما موضح في الشكل التالي:



الشكل 4-8 ادراج الحاوية HBox

5. استخدم نافذة عرض الخصائص Flex Properties ، لتغيير لون الخلفية ، موجودة بالاسفل من القسم Style.



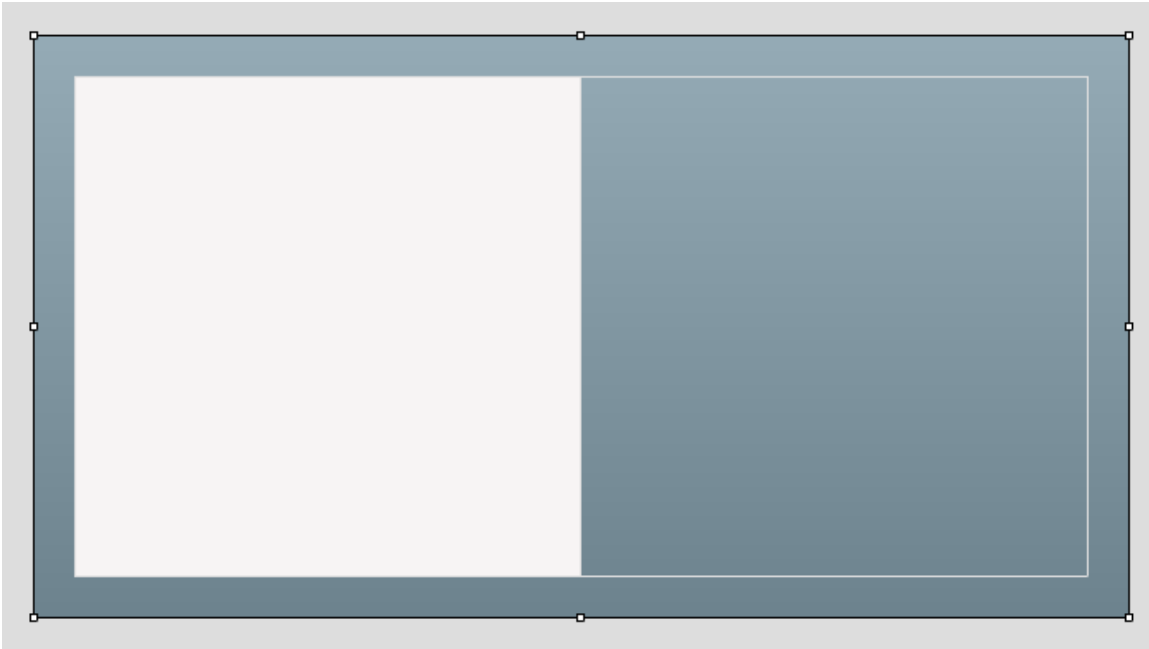
الشكل 4-9 تغيير لون الخلفية

6. قم بادراج حاوية من النوع VBox داخل الحاوية HBox التي تم انشاءها سابقا ، قم بضبط الارتفاع

%50 Height والعرض %100 Width .

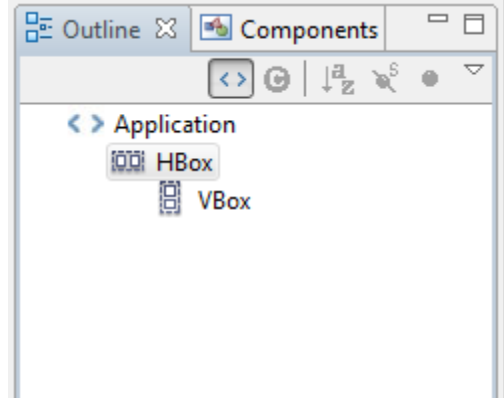
الحاوية HBox هي الحاوية الام للحاوية VBox

7. قم بتغيير لون خلفية الحاوية VBox .



الشكل 4-10 الحاوية VBox كحاوية داخلية للحاوية Hbox

8. افتح النافذة Outline View بجانب Component لمشاهدة نافذة عرض الخطوط العريضة . Outline View

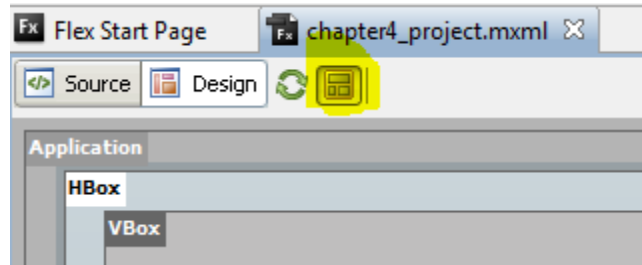


الشكل 4-11 نافذة عرض الخطوط العريضة

كما تلاحظ فان الحاوية VBox داخل الحاوية Hbox ، وان الحاوية Hbox بداخل الحاوية Application .

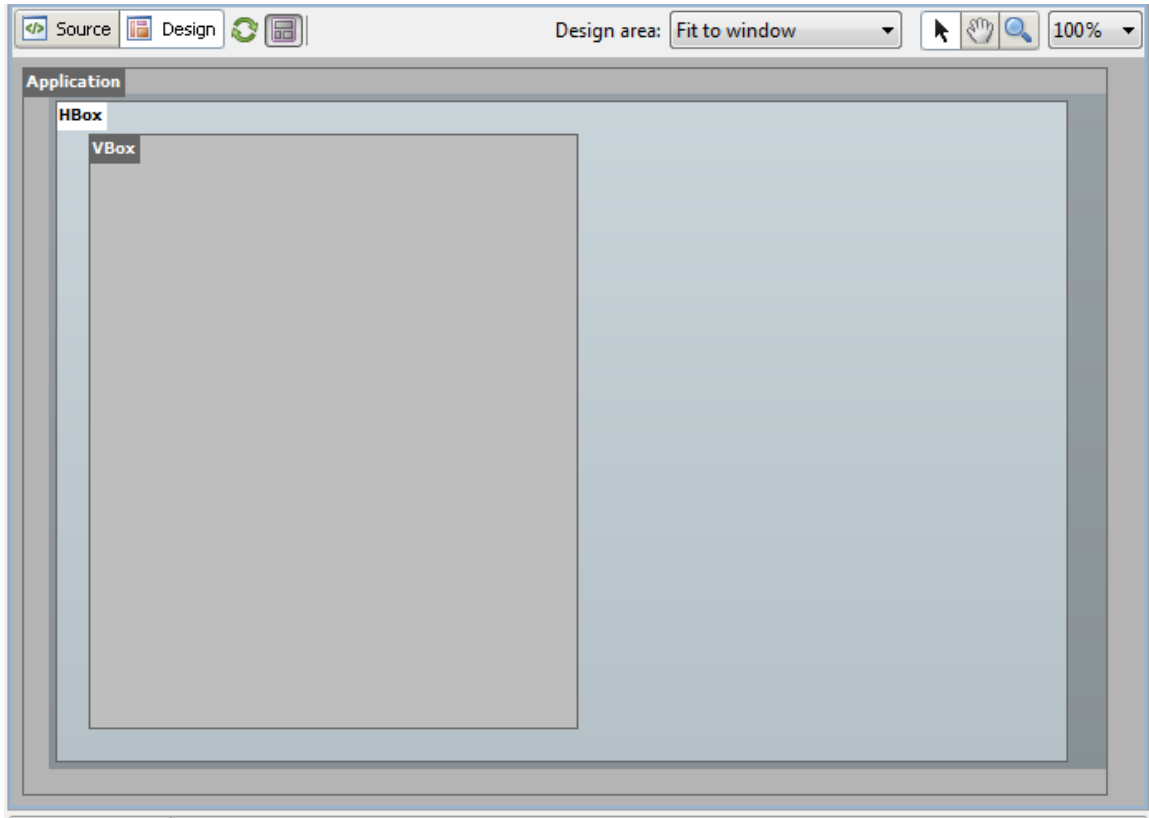
اذا نظرنا في الجزء العلوي من منصة العمل وبالتحديد يمين الازرار Source و Design ، ستلاحظ زر عرض محيط الحاوية Show Surrounding Containers.

9. اضغط على زر عرض محيط الحاوية Show Surrounding Containers او اضغط على المفتاح F4 من لوحة المفاتيح .



الشكل 4-12 عرض محيط الحاوية

عند الضغط على الزر Show Surrounding Containers ستلاحظ العلاقة بين الحاويات في منصة التصميم.



الشكل 4-13 العلاقة بين الحاويات

10. اضغط على الزر Show Surrounding Containers لإلغاء عرض محيط الحاويات

11. افتح نافذة عرض التعليمات البرمجية Source لعرض التعليمات البرمجية .

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="horizontal">
  <mx:HBox width="100%" height="100%">
    <mx:VBox width="50%" height="100%"
      backgroundColor="#F7F4F4">
    </mx:VBox>
  </mx:HBox>
</mx:Application>
```

: Form Container

عندما كنا نتعامل مع الحاويات من النوع Hbox و VBox ، لا يمكن تغيير الخاصية Layout بحكم تعريفها ، في حين ان هذا يسعدنا كثيرا في تنظيم المحتويات ، الا انها تحتوي على بعض المشاكل . دعنا نشاهد مثال على ذلك:

1. افتح نافذة عرض التصميم Design Prespective.
2. في النافذة Component View ، افتح القسم Controls
3. ادرج الكائن Label في الحاوية Hbox
4. قم بتغيير نص الكائن Label الي First Name:
5. قم بادراج الكائن InputText



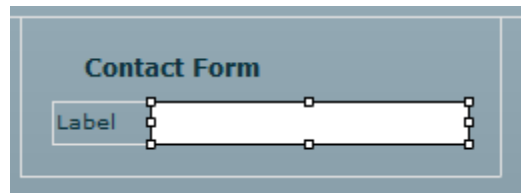
الشكل 14-4 الكائنات داخل الحاوية HBox

- بغض النظر عن المكان الذي وضعت فيه الكائنات يقوم مدير التنسيق Layout Manager بتنظيم الكائنات تلقائيا بشكل افقي ، وقد تكون هذه النتيجة غير مرغوب فيها.
6. قم بحذف الكائنات داخل الحاوية Hbox .
 7. اذهب الي القسم Layout في النافذة Components View ، قم بادراج الحاوية Form في الحاوية Hbox .
- تسمح لنا الحاوية Form بتنظيم وتنسيق محتويات النموذج بسهولة
8. قم بإدراج الكائن Form Heading من القسم Layout داخل الحاوية Form التي قمنا بوضعها سابقا داخل الحاوية Hbox .
 9. قم بتغيير النص الي Contact Form كما موضح في الشكل التالي :



الشكل 15-4 الكائن FormHeading

10. قم بادراج الكائن TextInput داخل الحاوية Form تحت الكائن Form Heading.



الشكل 16-4 ادراج الكائن TextInput

- نلاحظ انه تم ادراج الكائن Label مع الكائن TextInput ، الحاوية Form تعالج ذلك تلقائيا .
11. اضغط Double Click على الكائن Label لتغيير النص الي : First Name .
12. قم بإضافة اثنين كائن مرة اخرى من النوع TextInput للاسم الاخير : Last Name و البريد الالكتروني : Email .

The screenshot shows a window titled "Contact Form" with a light blue background. It contains three vertically stacked input fields. Each field has a label on the left and a white text box on the right. The labels are "First Name :", "Last Name :", and "E-mail :".

الشكل 4-17 الكائنات داخل الحاوية Form

13. قم بإضافة الكائن Button كما قمنا في السلبق بإضافة الكائن TextInput . لاحظ المشكلة التي حدثت .

The screenshot shows the same "Contact Form" window as before, but with an additional row at the bottom. This row contains a "Label" and a "Button" side-by-side. The "Label" is a small grey box with the text "Label", and the "Button" is a slightly larger grey box with the text "Button".

الشكل 4-18 اضافة الكائن Button

- تمت اضافة الكائن Label ونحن لا نحتاج الي ذلك ، هنالك طريقتين لحل هذه المشكلة ، ابسط طريقة نقوم بضغط Double Click على الكائن Label وحذف النص الموجود ، الطريقة الثانية نقوم بادراج الكائن Button بجانب الكائن TextInput .
14. قم بفتح نافذة عرض التعليمات البرمجية

```
<mx:HBox width="100%" height="100%">
  <mx:VBox width="50%" height="100%"
  backgroundColor="#F7F4F4">
    </mx:VBox>
    <mx:Form>
      <mx:FormHeading label="Contact Form"/>
      <mx:FormItem label="First Name :">
```

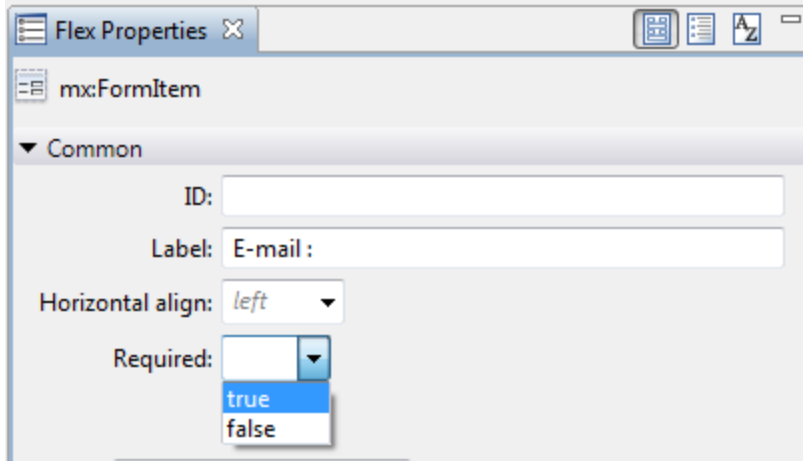
```

        <mx:TextInput/>
    </mx:FormItem>
    <mx:FormItem label="Last Name :">
        <mx:TextInput/>
    </mx:FormItem>
    <mx:FormItem label="E-mail :">
        <mx:TextInput/>
        <mx:Button label="Button"/>
    </mx:FormItem>
</mx:Form>
</mx:HBox>

```

الحاوية Form تقوم بتنظيم الكائنات وذلك بوضع كل كائن في حاوية اخرى تسمى FormItem ولها خاصية تسمى Label ، لذلك لا نحتاج الي ادراج الكائن Label ، دعنا ننتقل الي ميزة اخرى من مميزات الحاوية Form .

15. قم باختيار الكائن E-mail من عناصر الحاوية Form ، اذا نظرت الي نافذة الخصائص Flex Properties ستلاحظ قائمة منسدلة بجوار الكلمة Required في القسم Common .



الشكل 19-4 الخاصية Required

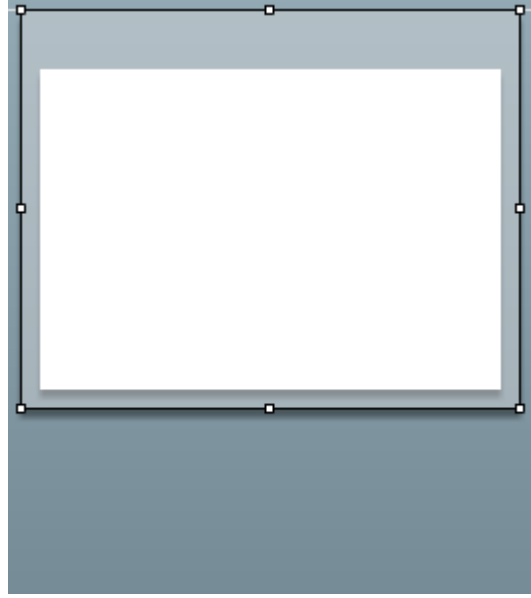
اذا اخترت القيمة true للخاصية Required ، سيتم اضافة علامة نجمية تلقائيا للكائن ، لاحقا سنتحدث عن كيفية معالجة الخاصية Required ومعالجة عملية التحقق برمجيا.

:Panel Container

هذه الحاوية فريدة من نوعها شبيهة جدا بحاوية التطبيقات ويمكن ان تتعامل مع كل انماط التنسيق الثلاثة (Absolute – Horizontal – Vertical) ، وهذا ما يعطينا درجة من المرونة كما سنلاحظ ذلك ، سنقوم بإعادة إنشاء نموذج الاتصال Contact Form بإستخدام الحاوية Panel .

1. سنقوم بحذف الحاوية Form وكل محتوياتها

2. نقوم بوضع حاوية Panel مكانها



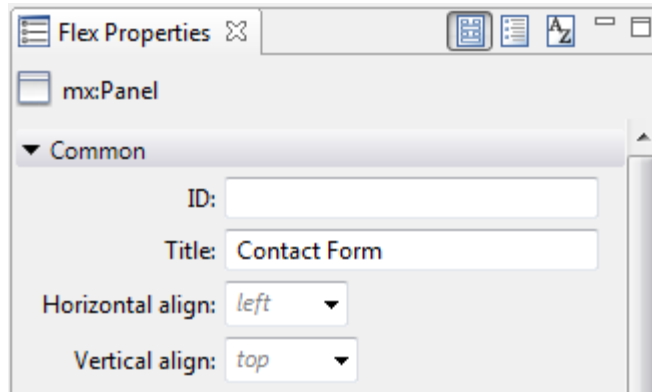
الشكل 20-4 الحاوية Panel

تحتوي الحاوية Panel على حدود نصف شفافة يمكن كتابة عنوان للوحة Panel من خلال الخاصية title ، عند ادراج الحاوية Panel نلاحظ عدم ظهور صندوق الحوار الخاص بضبط حجم الحاوية كما في الحاوية VBox و HBox .

3. من نافذة خصائص فيليكس Flex Properties قم باعادة ضبط عرض الحاوية width الي 50%

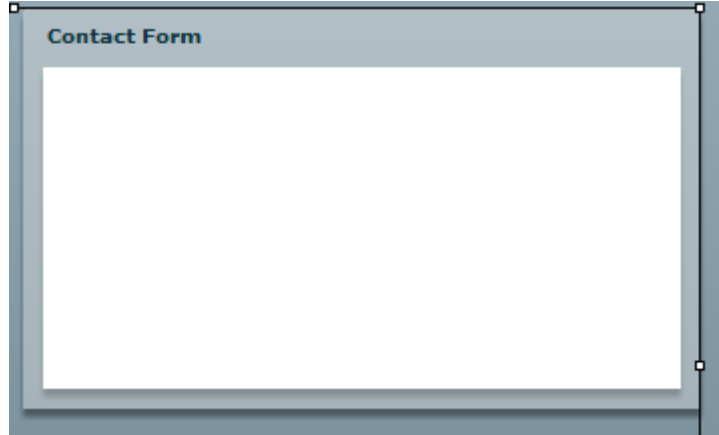
اذا نظرنا الي القسم Common من نافذة Flex Properties سنلاحظ صندوق ادخال Title لادخال عنوان الحاوية .

4. سنقوم بكتابة النص Contact Form في الحقل Title كما موضح في الشكل التالي :



الشكل 21-4 الحقل Title

بعد كتابة العنوان في الحقل Title اضغط على المفتاح Enter من لوحة المفاتيح ،



الشكل 22-4 العنوان في الحاوية Panel

المنصة الخالية باللون الابيض هي القسم الذي يتم فيه وضع المحتويات،

5. سنقوم بضبط نمط التنسيق الي النمط Vertical، ويمكن ان نقوم بذلك من خلال نافذ Flex Properties كما في Application Container.

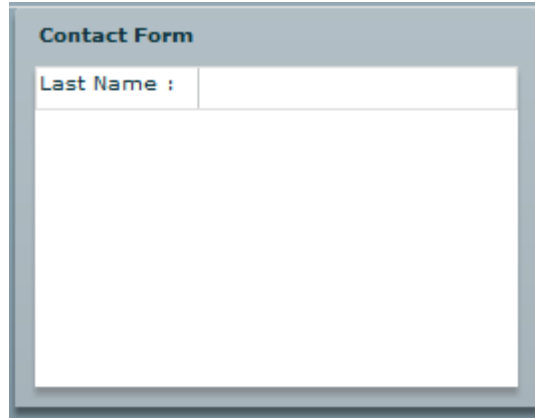
في هذا المثال يكون تنسيق النموذج (Form) مختلف قليلا من النموذج السابق ، لاننا نحتاج الي الجميع بين نمطين مختلفين من الانماط VBox و HBox ، سنقوم بتجرب ذلك من خلال المثال التالي.

6. سنقوم بإدراج HBox داخل المنصة البيضاء في الحاوية Panel .

7. قم بضبط كل من العرض والارتفاع <fit to content> وذلك بحذف كل القياسات التي تظهر في صندوق حوار اعدادات الحجم، وذلك يتم ترتيب المحتويات داخل لنموذج من الشمال الي اليمين

8. سنقوم بإدراج Label و TextInput داخل الحاوية HBox .

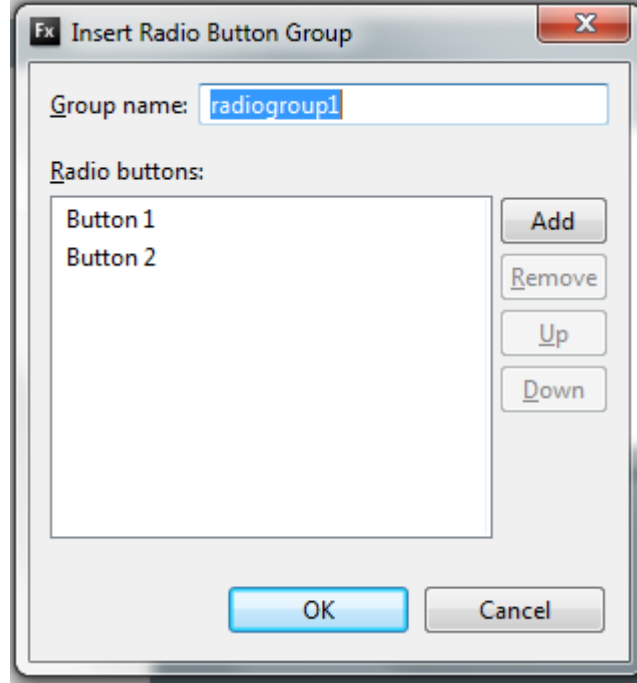
9. سنقوم بتعديل الخاصية Text في الكائن Label الي : Last Name .



الشكل 23-4 المحتويات داخل الحاوية Panel

الآن هناك فرصة جيدة لاستخدام كائنات أخرى داخل النموذج Form ، RadioButton . كما في لغة HTML الكائن RadioButton في فيليكس لا يمكن إختيار أكثر من عنصر في نفس الوقت وهو عكس تماما للكائن CheckBox ، ولتحقيق ذلك بشكل سليم يجب وضع كل الكائنات من النوع RadioButton معا في حاوية واحدة تسمى RadioButtonGroup .

10. سنقوم بادراج الحاوية RadioButtonGroup تحت الحاوية HBox في الحاوية panel



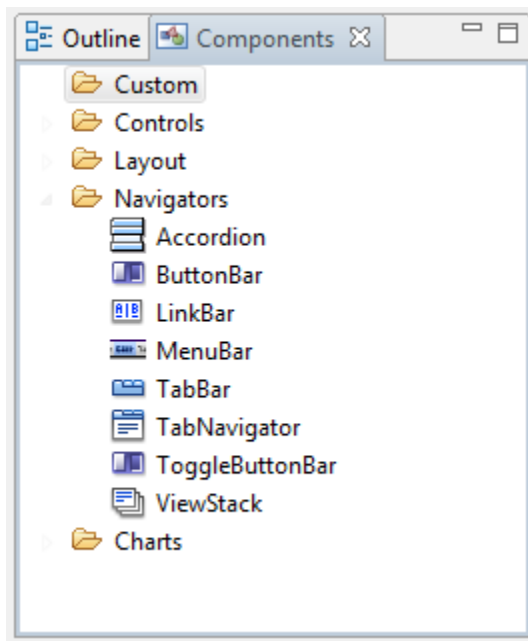
الشكل 4-24 صندوق حوار ادراج RadioButtonGroup

11. سنقوم بتغيير Group name الي ContactType
12. اضغط على Button 1 وسنقوم بتغيير النص الي Telephone
13. اضغط على Button 2 وسنقوم بتغيير النص الي E-mail
14. اضغط على ازر Add لاضافة زر ثالث ، سيظهر صندوق حوار يسمح لنا بكتابة اسم الزر الجديد سنكتب No Contact .

الشكل 4-25 اضافة الكائن RadioButton

:Navigation Container

خلافًا لصفحات HTML انك لا تذهب في كل مرة الي صفحة جديدة عند فتح احد الرابط ، بدلا من ذلك ، كل المحتويات الخاصة بموقعك تكون في ملف swf واحد ويمكن الوصول اليها عند اختيار الرابط المناسب ، سيكون هذا واضحا عندما نتحدث عن الحالات (States) .
 ككل التطبيقات دائما ما نسعى الي توفير وسيلة سهلة للتنقل بين محتويات الصفحة او الموقع ، فيليكس يساعدنا في ذلك ، حيث يوفر لنا حاويات مخصصة لهذا الغرض ، وهذه الحاويات تسمى Navigator Container ويمكن استخدامها من خلال نافذة المكونات Component View .

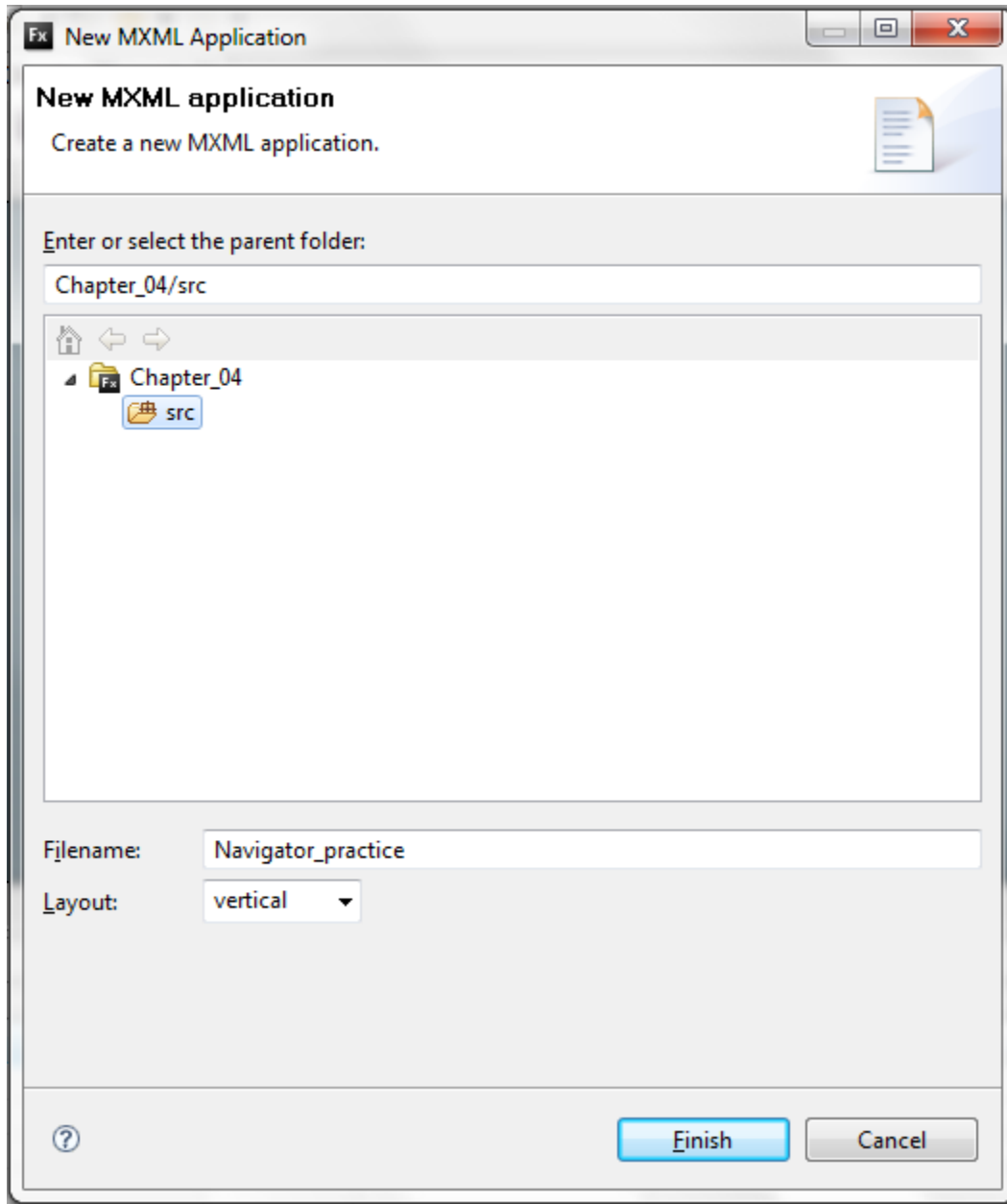


الشكل 4 – 25 الحاوية Navigators

الحاوية **ViewStack Container** :

نحن الان بصدد عمل تطبيق بسيط جدا حيث يمكننا من فهم الية عمل Navigators .

1- من قائمة File اختر New < MXML Application



شكل 4-26 صندوق حوار انشاء تطبيق MXML جديد

2- سنقوم بتسمية تطبيق MXML (Navigator_practice) .

3- نجعل تخطيط الصفحة Layout (vertical) .

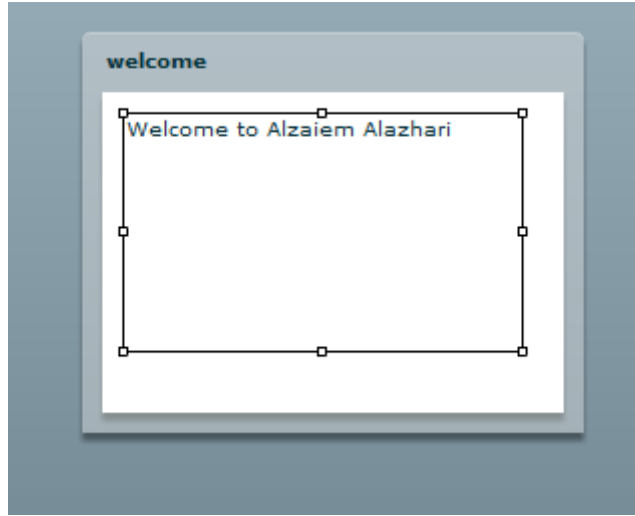
4- سنقوم بادراج حاوية من النوع Panel في منصة العمل ، ونضع لها عنوان Title فليكن . Welcome

5- سنقوم بضبط حجم الحاوية وذلك بجعل العرض 250 width والارتفاع 200 Height . قبل هذه النقطة لم نتحدث بشكل كاف عن الخاصية Id ، من المهم جدا اعطاء كل مكون من مكونات التطبيق تعريف فريد ليتمكن الوصول اليه بسهولة بواسطة MXML او ActionScript ، وبما ان MXMK و ActionScript حساسة لحالة الاحرف يجب اتباع قواعد تسمية المتغيرات عند تعريف الخاصية ID .

6- من نافذة الخصائص في فيليكس نقوم بوضع تعريف للحاوية Panel وليكن welcome .

7- سنقوم بادراج المكون text من نافذة المكونات Component View داخل الحاوية Panel .

8- سنقوم بكتابة نص داخل المكون Text كما موضح في الشكل التالي :

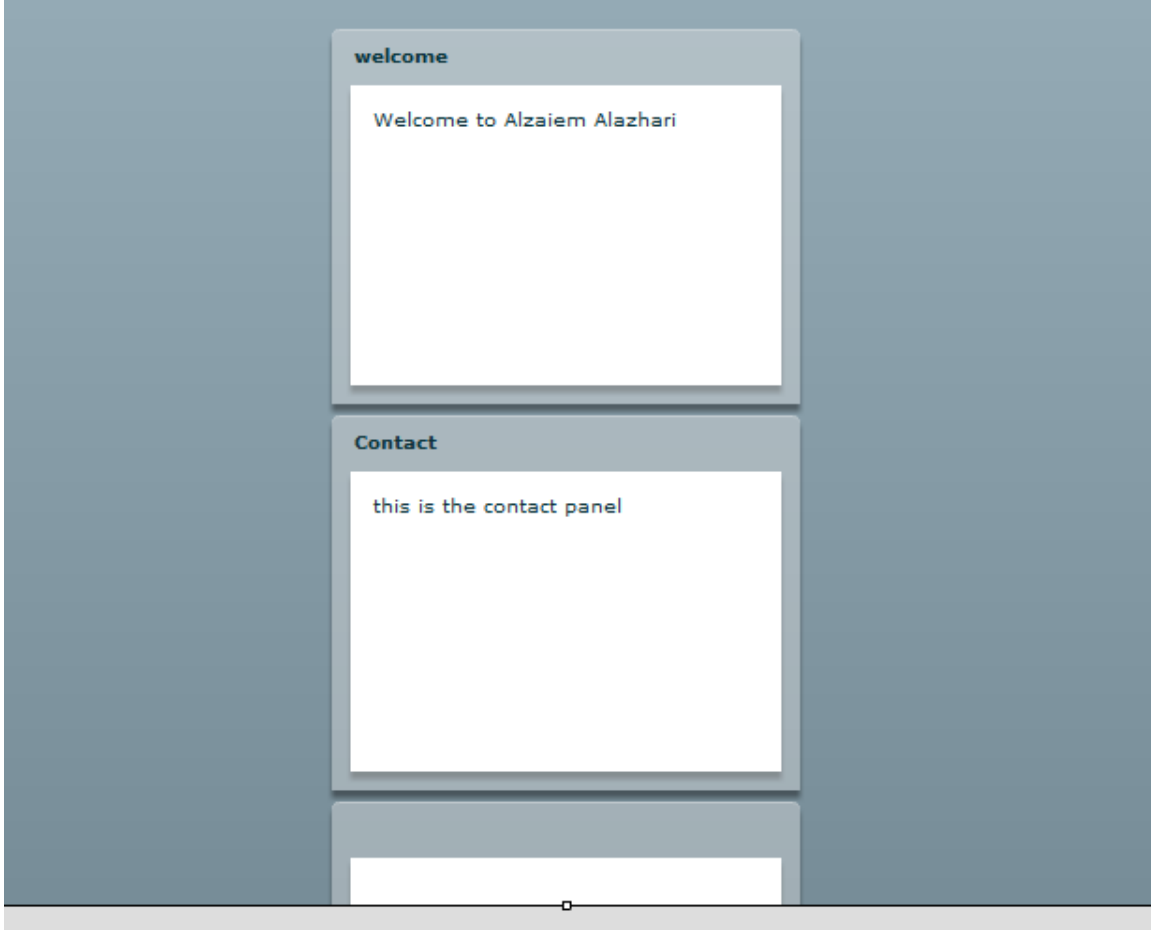


شكل 4 – 27 Welcome Panel

9- سنقوم بادراج حاوية اخرى من النوع Panel ، وبما ان تخطيط الصفحة Vertical توضع تقائيا اسفل الحاوية welcome Panel ، سيكون عنوان (Title) هذه الحاوية Contact وكذلك تعريفها (ID) .

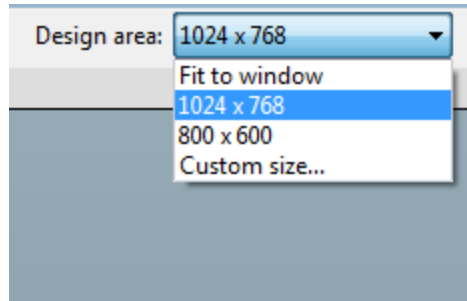
10- سنقوم بادراج المكون text على الحاوية Contact Panel .

11- سنقوم الان بادراج حاوية اخرى من النوع Panel ونلاحظ ان هذه الحاوية جزء منها خارج اطار الصفحة .



شكل 4 – 28 الحاوية Panel خارج منصة التصميم

اذا كنا نرغب في مشاهدة كل الحاويات بالشكل الكامل ، نذهب اعلى منصة التصميم هناك قائمة منسدلة تسمى Design Area لاعداد حجم منصة التصميم .

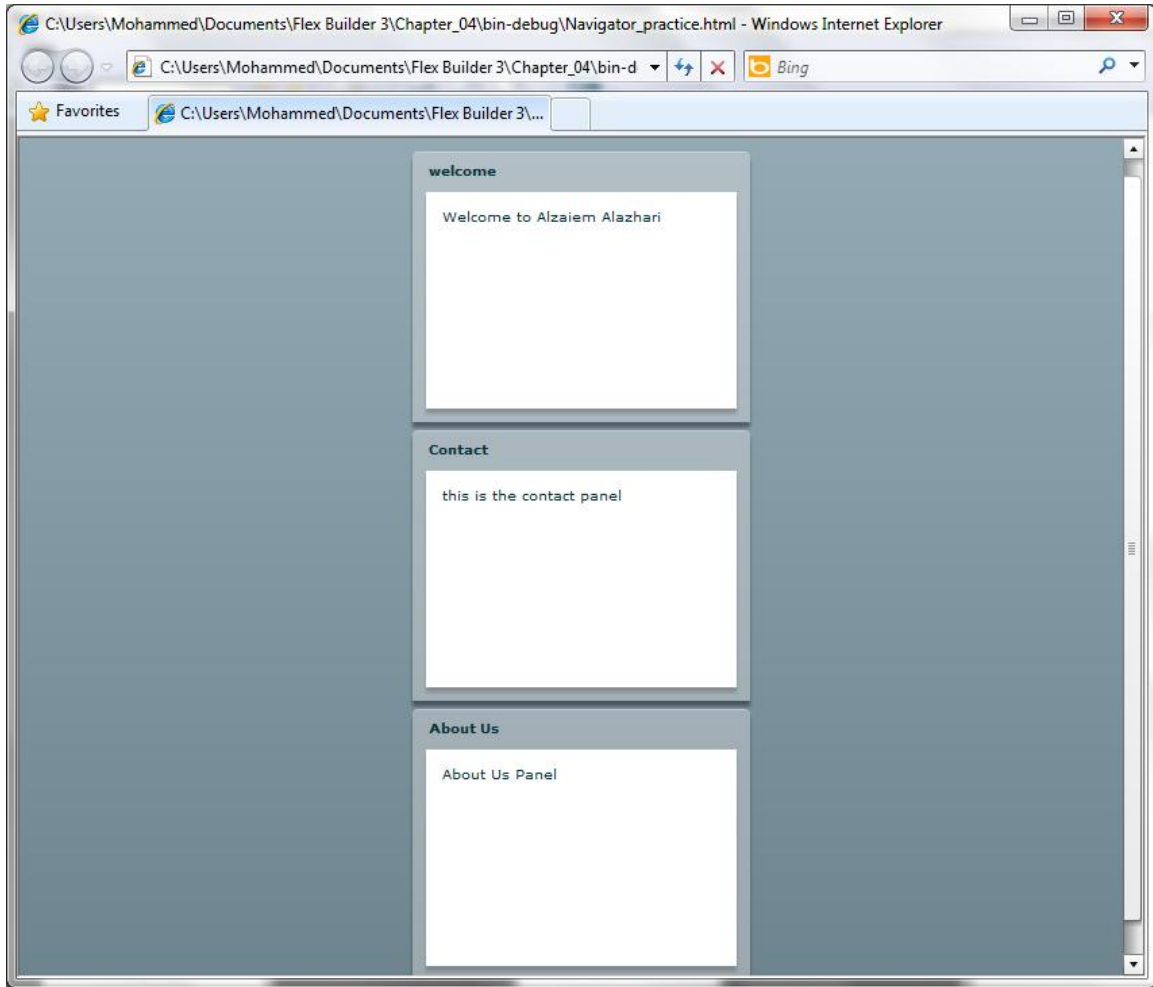


شكل 4 – 29 ضبط حجم منصة التصميم

12- سنقوم باعادة ضبط حجمة منصة التصميم الي 1024 X 768 ، الان يمكنك استخدام الشريط

scrollbars لمشاهدة بقية محتويات الصفحة .

13- الان سنقوم بتنفيذ التطبيق .



شكل 4 - 29 شكل التطبيق في الوقت الحالي

14- ننتقل الي نافذة عرض التعليمات البرمجية Source لنشاهد التعليمات التي تم انشاءها .

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">
  <mx:Panel width="250" height="200"
    layout="absolute" title="welcome" id="welcome">
    <mx:Text x="10" y="10"
      text="Welcome to Alzaiem Alazhari"
      width="200" height="120" enabled="true"/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="contact" title="Contact">
    <mx:Text x="10" y="10" width="200"
      height="120" text="this is the contact panel "/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="about" title="About Us">
```

```
<mx:Text x="10" y="10" text="About Us Panel "
width="200" height="120"/>
</mx:Panel>
```

```
</mx:Application>
```

-15 قبل التعليمة البرمجية لاول Panel قم باجراج التعليمة البرمجية <mx:ViewStack>

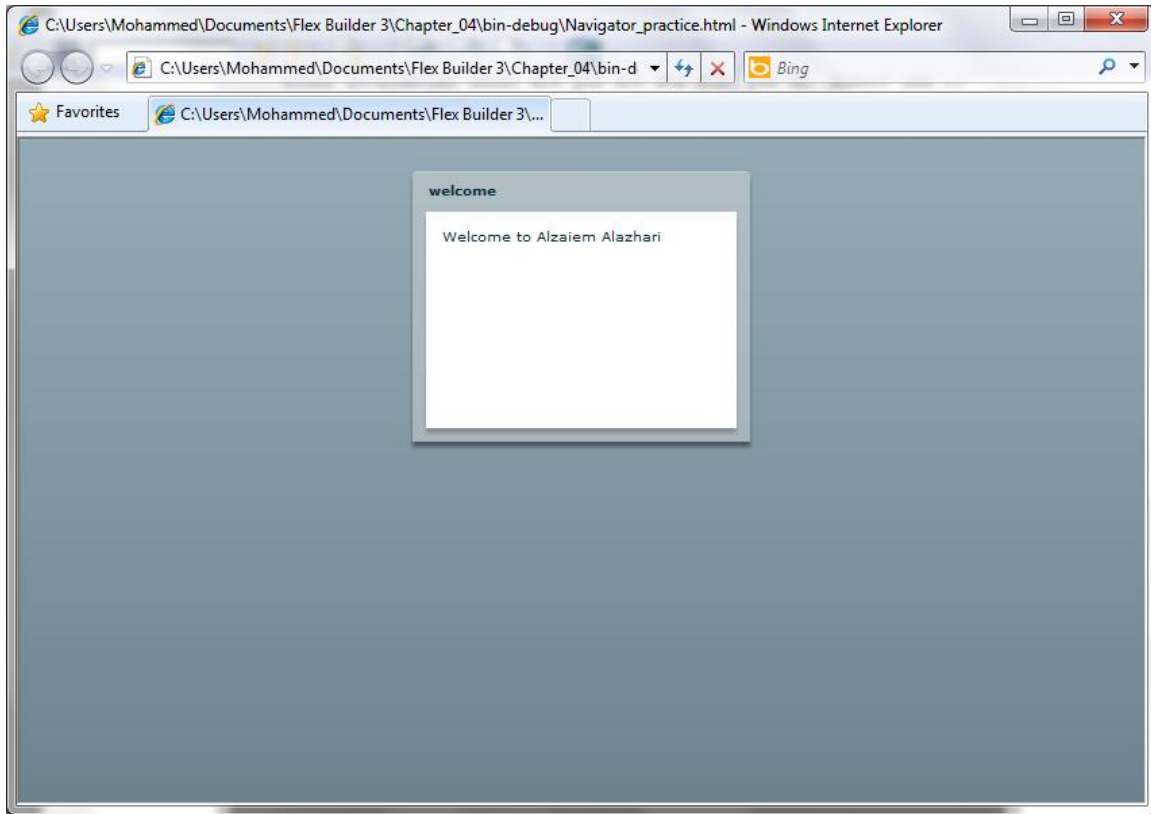
-16 بعد التعليمات البرمجية لآخر Panel قم باضافة التعليمة </mx:ViewStack> ، سيصبح

الشكل النهائي للتعليمات البرمجية كما موضح في التعليمات التالية :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">
```

```
<mx:ViewStack>
  <mx:Panel width="250" height="200"
    layout="absolute" title="welcome" id="welcome">
    <mx:Text x="10" y="10"
      text="Welcome to Alzaiem Alazhari"
width="200"
      height="120" enabled="true"/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="contact" title="Contact">
    <mx:Text x="10" y="10" width="200"
      height="120" text="this is the contact panel"/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="about" title="About Us">
    <mx:Text x="10" y="10" text="About Us Panel "
      width="200" height="120"/>
  </mx:Panel>
</mx:ViewStack>
</mx:Application>
```

-17 الان سنقوم بتشغيل التطبيق ، سنشاهد شكل مختلف .



شكل 4 – 30 الحاويات Panel داخل الحاوية ViewStack

18- سنقوم باضافة الخاصية ID للحاوية ViewStack كما في الشكل التالي :

```
<mx:ViewStack id="myPage">
```

19- سنقوم باضافة الخاصية Label الي كل الحاويات Panel ،

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical">
<mx:ViewStack id="myPage">
  <mx:Panel width="250" height="200"
    layout="absolute" title="welcome" id="welcome"
label="Welcome">
    <mx:Text x="10" y="10"
      text="Welcome to Alzaiem Alazhari"
width="200"
      height="120" enabled="true"/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="contact" title="Contact"
label="Contact Us">
    <mx:Text x="10" y="10" width="200"
      height="120" text="this is the contact panel"/>
  </mx:Panel>
</mx:ViewStack>
</mx:Application>
```

```

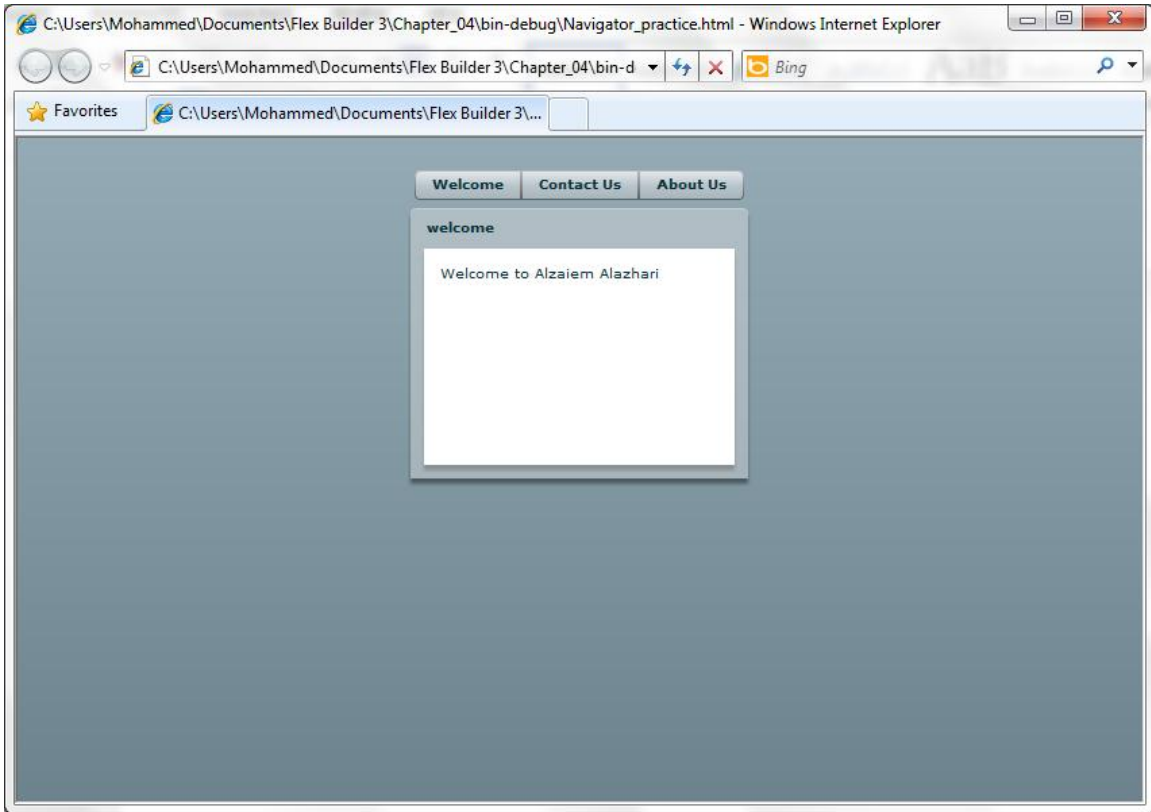
</mx:Panel>
<mx:Panel width="250" height="200"
    layout="absolute" id="about" title="About Us"
label="About Us">
    <mx:Text x="10" y="10" text="About Us Panel "
        width="200" height="120"/>
</mx:Panel>
</mx:ViewStack>
</mx:Application>

```

20- سنقوم بوضع التعليمات البرمجية التالية قبل التعليمات البرمجية للحاوية ViewStack

```
<mx:ButtonBar dataProvider="{myPage}"/>
```

21- سنقوم بتشغيل التطبيق الان ،

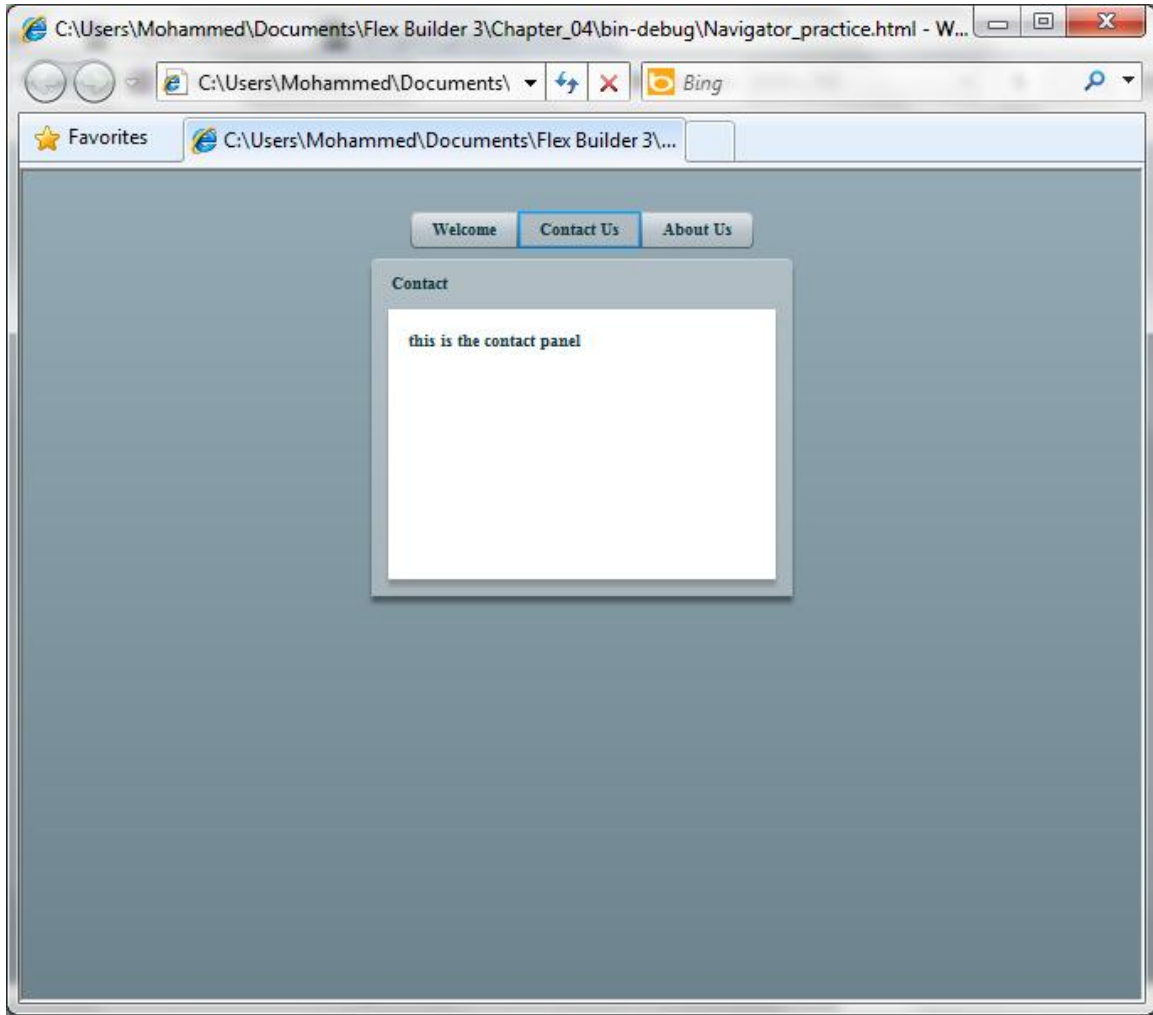


شكل 4 – 31 الشكل النهائي للتطبيق باستخدام الحاوية ViewStack

يستخدم حاوية فقط وبعض الخصائص والمكون ButtonBar حصلنا على تطبيق كامل الوظائف ، مع ذلك فمرونة فيليكس لا تتوقف عند هذا الحد .

22- نرجع الي نافذة عرض التعليمات البرمجية ونقوم بتغيير ButtonBar الي ToggleButtonBar

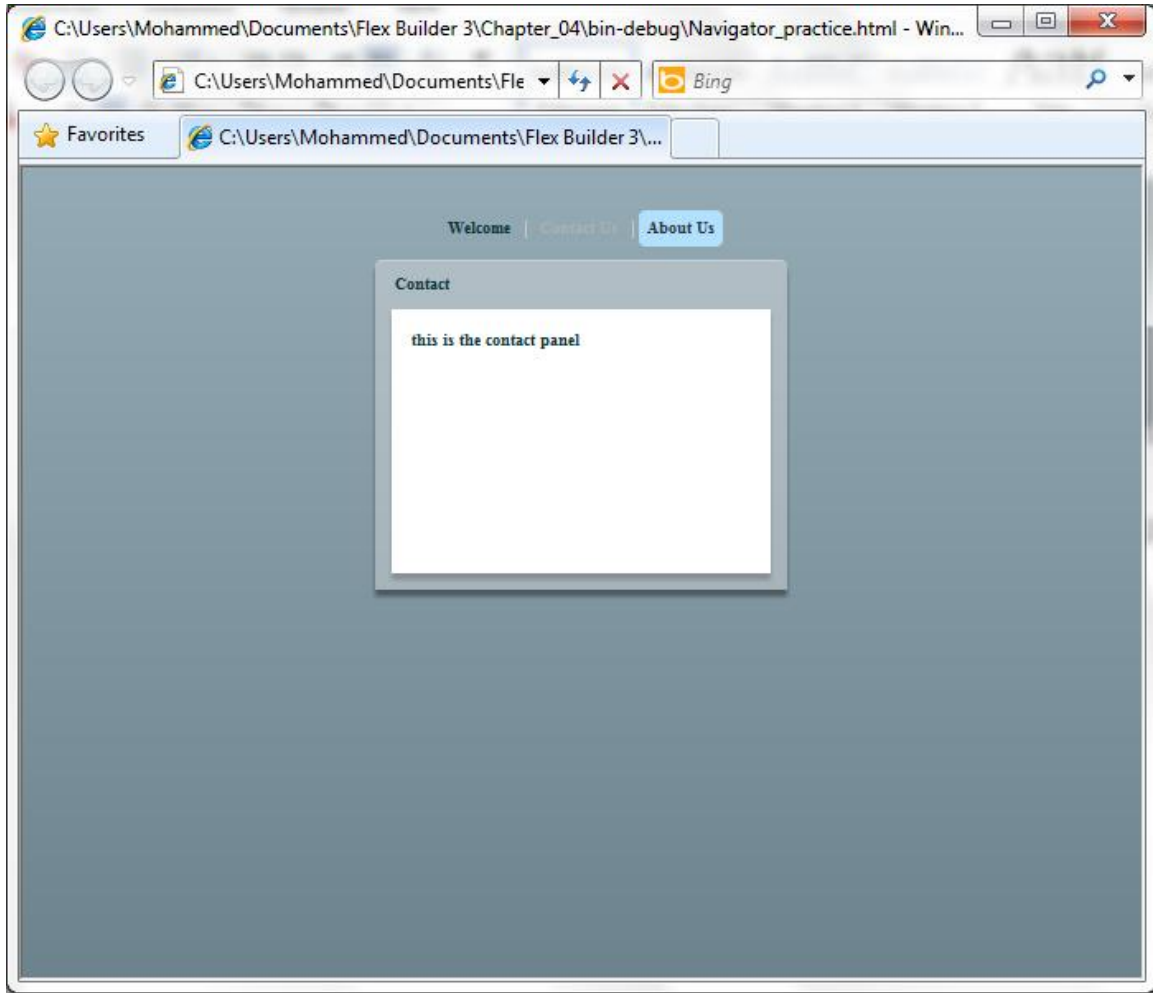
```
<mx:ToggleButtonBar dataProvider="{myPage}"/>
```



شكل 4 – 32 شكل التطبيق باستخدام ToggleButtonBar

23- سنقوم بتغيير ToggleButtonBar الي LinkBar

```
<mx:LinkBar dataProvider="{myPage}"/>
```



شكل 4 – 33 شكل التطبيق باستخدام LinkBar

عمل Navigation باستخدام ActionScript :

من الممكن استخدام تعليمات ActionScript لاجراء عملية Navigation ، سنقوم الان بعمل سيناريو بسيط لشرح ذلك .

1- سنبدأ بإنشاء حاوية من النوع HBox بعد اغلاق الوسم ViewStack ، وثلاثة مكونات من النوع Button بداخلها كما في المثال التالي :

```
</mx:ViewStack>
<mx:HBox>
  <mx:Button label="Welcome" />
  <mx:Button label="Contact Us" />
  <mx:Button label="About Us" />
</mx:HBox>
</mx:Application>
```

2- الان سنقوم بتشغيل التطبيق ، يجب ان تظهر الازرار الثلاثة اسفل الحاوية View Stack .

الحاوية ViewStack لها خاصية تسمى SelectedIndex ، هذه الخاصية تسمح لنا باستدعاء اي عنصر داخل الحاوية بواسطة بواسطة الفهرس الخاص به ، مثلا لاستدعاء الحاوية welcome باعتبارها اول عنصر داخل الحاوية ViewStack نستخدم الامر

```
myPage.selectedIndex=0
```

3- سنستخدم التعليمة البرمجية اعلاه مع المكون Button كما في المثال التالي :

```
<mx:HBox>
  <mx:Button label="Welcome"
    click="myPage.selectedIndex=0"/>
  <mx:Button label="Contact Us"
    click="myPage.selectedIndex=1"/>
  <mx:Button label="About Us"
    click="myPage.selectedIndex=2"/>
</mx:HBox>
```

4- الان سنقوم بتشغيل التطبيق ، نلاحظ انه يعمل بصورة جيدة مع الازرار الجديدة .
يمكننا استخدام خاصية اخرى بدلا من الخاصية SelectedIndex هذه الخاصية تسمى SelectedChild وعند استخدام هذه الخاصية يتم استدعاء العنصر بواسطة المعرف Id بدلا من الفهرس

5- الان سنقوم بتعديل التطبيق كما في المثال التالي :

```
<mx:HBox>
  <mx:Button label="Welcome"
    click="myPage.selectedChild=welcome"/>
  <mx:Button label="Contact Us"
    click="myPage.selectedChild=contact"/>
  <mx:Button label="About Us"
    click="myPage.selectedChild=about"/>
</mx:HBox>
```

6- الان سنقوم بتشغيل التطبيق ،

الحاوية TabNavigator والحاوية Accordion :

تقوم الحاوية TabNavigator والحاوية Accordion بدمج جانب التصميم مع Navigator ، وهي سهلة الاستخدام .

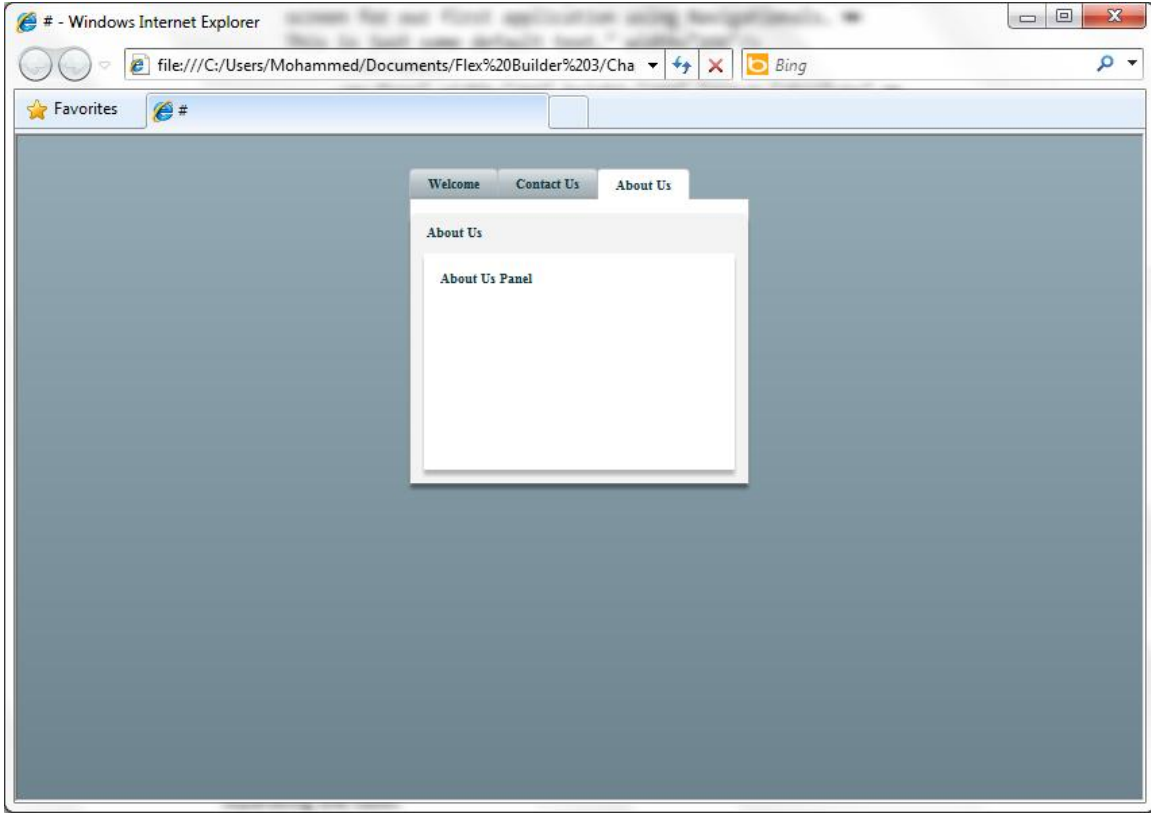
لمشاهدة مثال على ذلك ، سنقوم بتعديل بعض التعليمات البرمجية في التطبيق السابق .

1- سنقوم بحذف الوسم LinkBar والحاوية HBox التي تم انشاءها في المثال السابق .

2- سنقوم بتغيير الوسم ViewStack الي TabNavigator ، يجب ان يكون شكل التطبيق كما موضح في المثال التالي :

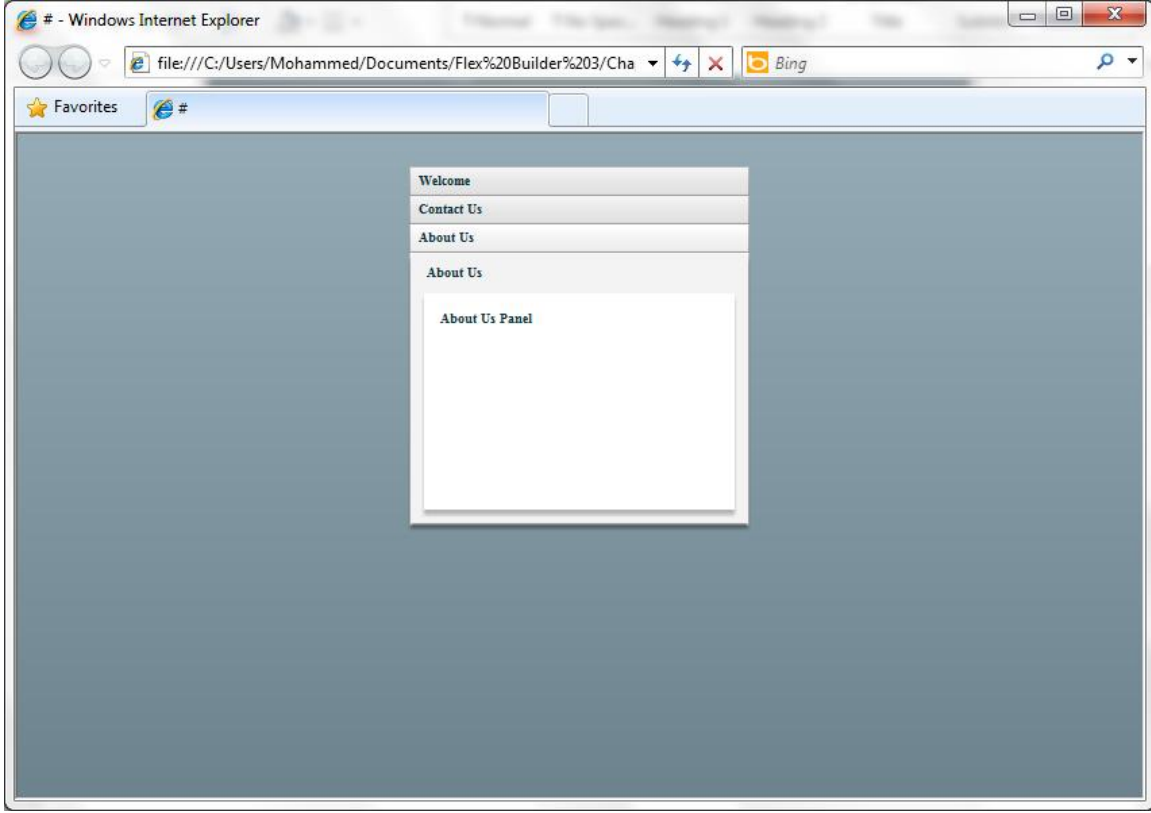
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="vertical" fontWeight="bold" fontFamily="Times New
Roman">
<mx:TabNavigator id="myPage">
  <mx:Panel width="250" height="200"
    layout="absolute" title="welcome" id="welcome"
label="Welcome">
    <mx:Text x="10" y="10"
      text="Welcome to Alzaiem Alazhari"
width="200"
      height="120" enabled="true"/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="contact" title="Contact"
label="Contact Us">
    <mx:Text x="10" y="10" width="200"
      height="120" text="this is the contact panel
    "/>
  </mx:Panel>
  <mx:Panel width="250" height="200"
    layout="absolute" id="about" title="About Us"
label="About Us">
    <mx:Text x="10" y="10" text="About Us Panel "
      width="200" height="120"/>
  </mx:Panel>
</mx:TabNavigator>
</mx:Application>
```

3- الان سنقوم بتشغيل التطبيق ، ويجب ان تكون النتائج كما في الشكل التالي :



شكل 4 – 34 الحاوية TabNavigator

4- سنقوم الان بتغيير الوسم TabNavigator الي Accordion .



شكل 4 – 35 الحاوية Accordion

الحالات States:

الان تغيرت وجهة نظرك تمام من خلال التدريبات السابقة في عملية التصفح (التنقل بين الصفحات)، لكن ماذا عن سيناريو نريد فيه تغير بعض المعلومات فقط.

دعنا نعود خطوة الي الوراء الي تصميم صفحات XHTML.

كما اشرنا سابقا ، تصميم مواقع الانترنت التقليدية غير SWf تتكون في العادة من عدد من صفحات XHTML التي يتم ترتيبها في شكل تسلسل هرمي مرتبطه ببعضها بواسطة نظام التصفح او التنقل عبر الصفحات. نظام التصفح هو عبارة عن سلسلة من الروابط التشعبية التي من خلالها يتنقل المستخدم من صفحة الي أخرى.

مع ان معظم صفحات الانترنت تعمل بهذا النظام ، الا انه غير فعال ، في كل مرة ينقر المستخدم على رابط ، يتم ارسال طلب الصفحة الي الخادم ، الخادم بدوره يقوم بتحديد الصفحة المطلوبة وارسالها الي العميل ليتم تحميلها على متصفح المستخدم.

تمت معالجة كل هذه القضايا بواسطة الفلاش من خلال استيعاب موقع انترنت كامل في ملف SWF واحد، الا في بعض الحالات المعقدة يمكن استخدام اثنين او ثلاثة ملفات SWF ودمجها في ملف SWF رئيسي.

كل هذا يحسن من كفاءة تصفح المواقع على شبكة الانترنت، حيث يتم ملف SWF مرة واحدة فقط ولا نحتاج الي الي ارسال طلب الي الخادم في كل مرة، شاهدنا في الأقسام السابقة طريقة واحدة وهي استخدام الحاويات في التصفح، هناك طريقة أخرى من خلال استخدام الحالة، التنقل من حالة الي حالة يشبه الانتقال من صفحة الي اخري، من خلال الأمثلة يمكن ان نفهم استخدام الحالات بصورة أفضل. من خلال هذه الأمثلة يمكننا مراجعة المفاهيم التي قمنا بمناقشتها سابقا، وكذلك سنتعلم بعض الأفكار الجديدة.

المثال الأول لدينا سيكون بسيط، سنقوم بإظهار اللوحات panels بالنقر على الروابط التشعبية.

1. انشئ تطبيق MXML جديد وذلك باختيار File > New > MXML Application
2. سمى التطبيق State_practice
3. إذا لزم الامر قم بتغيير التخطيط الي Absolute
4. اذا لزم الامر قم بالتبديل الي واجهة عرض التصميم.
5. اسحب الحاوية Panel الي الزاوية العلوية اليسرى من منصة التصميم كما موضح في الشكل التالي:



الشكل 4-36 موضع الحاوية Panel

6. قم بإعطاء الحاوية Panel عنوان وليكن Enemies of Ed، وذلك باستخدام نافذة خصائص فليكس او بالنقر المزدوج على راس الحاوية.

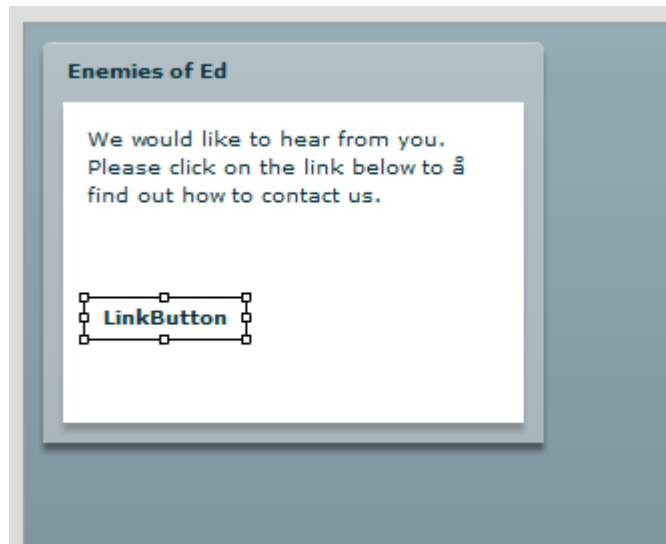
7. قم بسحب عنصر التحكم Text الي الساحة البيضاء من الحاوية Panel كما قمنا بذلك سابقا و اضع النص التالي:

We would like to hear from you. Please click on the link below to a
.find out how to contact us

8. اضغط Enter

انت تريد اظهار لوحة جديد عند حدوث حدث، لتحقيق ذلك نقوم بإنشاء حدث لتشغيل الحالة الجديدة، هذه فرصة جيدة لنقوم بتجربة عنصر تحكم لم نجربه من قبل، عنصر التحكم LinkButton. عنصر التحكم LinkButton هو اقرب الي الرابط التشعبي في XHTML

9. قم بسحب عنصر التحكم LinkButton الموجود ضمن فئة عناصر التحكم Controls من واجهة عرض المكونات ، ووضعه تحت عنر التحكم Text

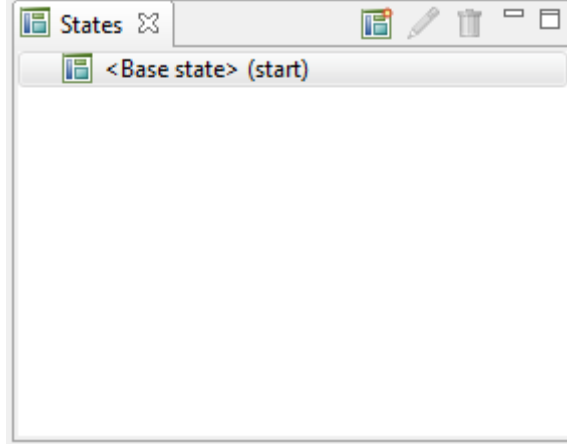


شكل 4-37 إضافة عنصر التحكم LinkButton الي الحاوية Panel

10. قم باختبار التطبيق لالقاء نظرة على عنصر التحكم LinkButton

تغيير الحالة :Changing State

الآن الخدعة التالية هي استخدام عنصر التحكم LinkButton لتغيير حالة التطبيق، لاحظ في الزاوية العلوية اليسرى من واجهة Flex Builder، أعلى نافذة عرض خصائص Flex Properties توجد نافذة عرض الحالات تسمى States كما موضح في الشكل أدناه.

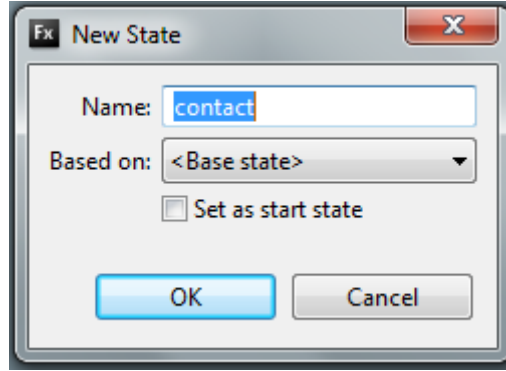


شكل 4-38 نافذة عرض الحالات

كل تطبيقات فليكس تبدأ بالحالة Base أو Start. وهذه هي الحالة الافتراضية أو بعبارة أخرى ما تراه في واجهة Flex Builder هو المخرج.

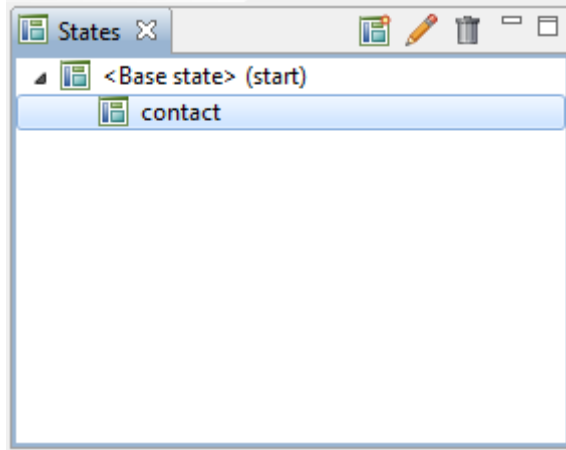
لكن الآن ستقوم بإضافة حالة إضافية

1. انقر بزر الماوس الأيمن على '<Base state>' في نافذة عرض الحالات وقم باختيار 'New State' أو انقر على زر 'New State' الموجود في الزاوية العلوية اليمنى لنافذة عرض الحالات، سيتم فتح صندوق حوار جديد يسمح لك بوضع اسم للحالة الجديدة.
2. سمي الحالة الجديدة 'contact' كما موضح في الشكل أدناه.



شكل 4-39 صندوق حوار إنشاء حالة جديدة

بناء على القائمة المنسدلة Based حيث يمكنك تحديد الحالة التي تريد إنشاء الحالة الجديدة عليها. وبما ان ليس لديك أي حالة أخرى نترك هذه المجموعة كما هي <Base state> ، يمكنك أيضا جعل هذه الحالة الجديدة كحالة افتراضية او الحالة التي يبدأ منها تشغيل التطبيق، اترك الخيار Set as start state غير محدد.
3. اضغط OK.



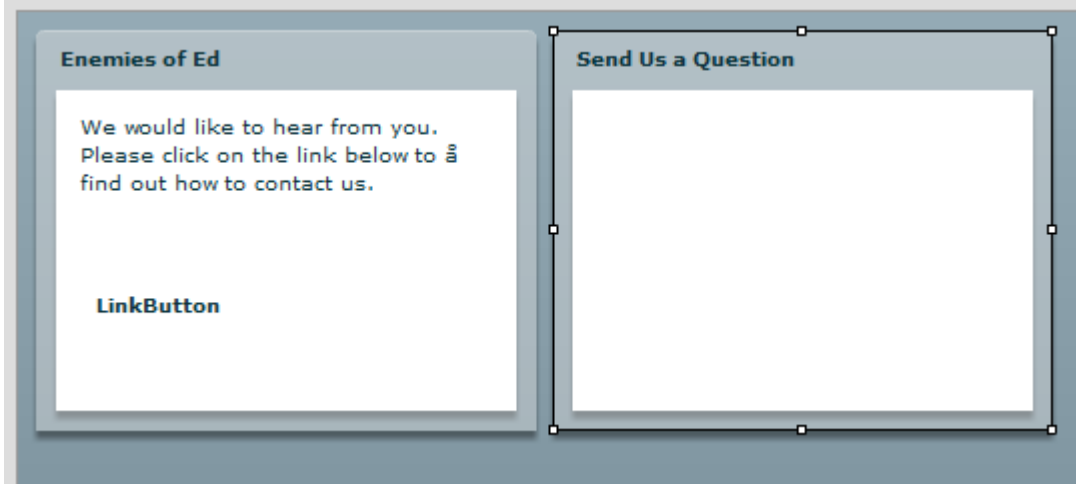
شكل 4-40 تمت إضافة الحالة الجديدة الي نافذة عرض الحالات

كل شيء يبدو كما هو في الحالة الجديدة ولكن ستقوم بتغييره ذلك، عند العمل على حالات متعدد من المهم جدا مراقبة الحالات في نافذة عرض الحالات لمعرفة أي الحالات نشطة الان، وهي الحالة المظلمة في نافذة عرض الحالات.

4. تأكد من ان الحالة contact هي التي تم اختيارها في نافذة عرض الحالات.

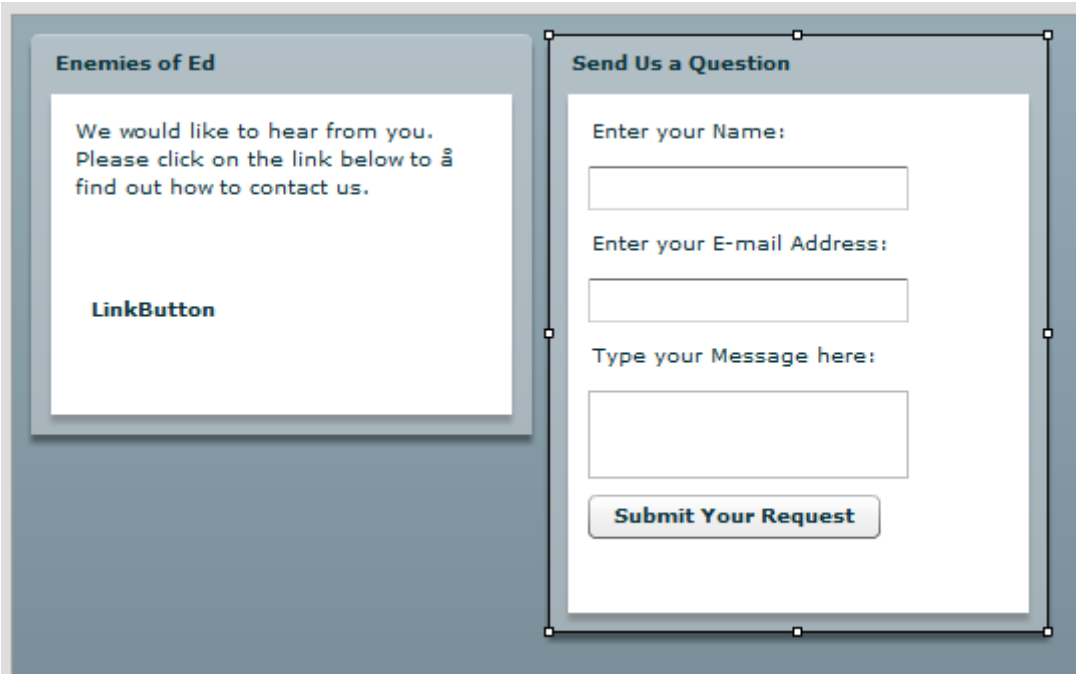
5. اسحب حاوية أخرى من النوع Panel على منصة التصميم، وقم بوضعها يمين الحاوية Panel

السابقة وقم باعطائها العنوان Send Us a Question.



شكل 4-41 تمت إضافة اللوحة الثانية الي الحالة contact

دعونا نقوم بانشاء نموذج بسيط لارسال بريد الكتروني كما موضح في الشكل ادناه، وبما ان النموذج لا يقوم بارسال رسائل بريد حقيقية ، تفاصيل النموذج غير مهمة.



شكل 4-42 النموذج contact

6. باستخدام الشكل أعلاه كدليل ، قم بادراج عناصر التحكم Label ، TextInput ، TextArea ، Button في الحاوية Panel.

7. في نافذة عرض الحالات انقر على الوسم <Base stat> ، ستختفي الحاوية Panel التي تمت اضافتها مؤخرا و اذا قمت بالنقر على الحالة contact تظهر الحاوية Panel الجديدة مرة أخرى. اذا قمت بتشغيل التطبيق الان، كل ما ستراه هو <Base state>. الان يجب ان تقوم بإضافة بعض التعليمات البرمجية لعناصر التحكم LinkButton و Button الموجودة في الحالة Base والحالة contact لتغير الحالة.

8. اذا لزم الامر انقر على الحالة Base في نافذة عرض الحالات.

9. انقر على عنصر التحكم LinkButton

10. في نافذة عرض خصائص فليكس اذهب الي الحقل On Click واكتب تعليمات الحدث On Click هنا كما تكتب في نافذة عرض التعليمات البرمجية

11. في الحقل On Click اكتب التالي:

```
currentState = 'contact'
```

علامة الاقتباس المفردة دلالة على ان السلسلة ضمن سلسلة أكبر، ألقى نظرة سريعة الي التعليمات البرمجية في نافذة عرض التعليمات البرمجية.

نحن الان نريد إعطاء عنصر التحكم Button الموجود في الحاوية Panel في الحالة contact بعض الوظائف.

12. اضغط على الوسم contact في نافذة عرض الحالات.

13. انقر في عنصر التحكم Button الموجود في الحاوية Panel.

14. في الحقل On Click اكتب التعليمة البرمجية التالية:

```
currentState = ''
```

15. علامة الاقتباس المفردة عندما تكون خالية دلالة على الحالة Base.

16. قم بحفظ وتشغيل التطبيق.

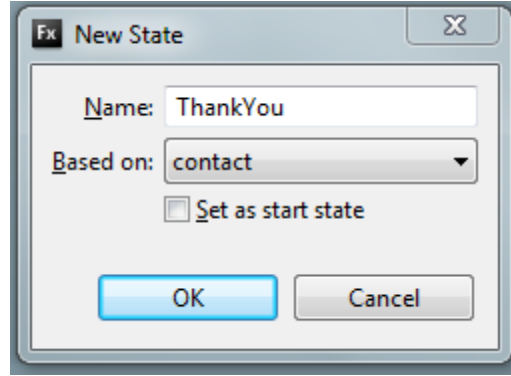
عند النقر على عنصر التحكم LinkButton، ينبغي ان يتم تشغيل الحاوية Panel الثانية، وعند النقر على عنصر التحكم Button بالحاوية الثانية، ينبغي ان تتم العودة الي الحاوية Panel الرئيسية والحالة Base Stae.

17. قم باختيار الحالة contact في نافذة عرض الحالات.

18. انقر على زر حالة جديدة مرة أخرى.

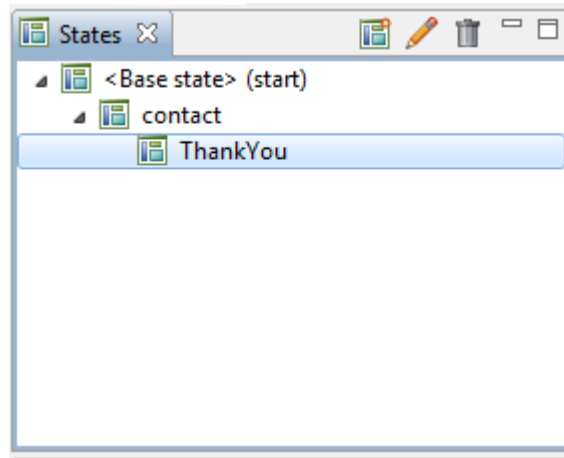
في هذه اللحظة تتساءل ما اذا كان بإمكانك إضافة حالة جديدة على الحالة contact، عند انشاء حلة جديد يجب عليك ان تحدد أي حالة تريد ان تكون الحالة الجديدة ضمنها، في حالتنا هذه ضمن الحالة contact.

19. قم بتسمية الحالة الجديدة ThankYou، كما موضح في الشكل ادناه، تذكر ذلك لا تستخدم أي مسافات عند التسمية.



شكل 4-43 صندوق حوار لحالة جديدة بنيت على الحالة contact

20. انقر على الزر Ok.



شكل 4-44 نافذة عرض الحالات مع التسلسل الهرمي للحالات

تلاحظ ان الحالة الجديدة التي تم انشاءها على الحالة contact تمت اضافتها الي نافذة عرض الحالات، هذا يجعل من السهل عرض التسلسل الهرمي الذي تم انشاءه من مختلف الحالات.

21. قم باختيار الحالة ThankYou، وقم بإدراج حاوية جديدة من النوع Panel في منصة التصميم.

22. قم باعطاء هذه الحاوية عنوان وليكن Thank you for Contacting us.

23. في جسم الحاوية ThankYou وباستخدام عنصر التحكم Text قم بإضافة النص التالي:

Thank you for sending us your inquiries. We will try to answer à your question in the next year or so.

24. قم بسحب عنصر التحكم Button الي الحاوية وبوضع النص Ok كدليل للزر Button ، كما موضح في الشكل ادناه.

The image shows a web form with three main sections. The top-left section is titled 'Enemies of Ed' and contains the text: 'We would like to hear from you. Please click on the link below to find out how to contact us.' Below this text is a button labeled 'LinkButton'. The top-right section is titled 'Send Us a Question' and contains three input fields: 'Enter your Name:', 'Enter your E-mail Address:', and 'Type your Message here:'. Below these fields is a button labeled 'Submit Your Request'. The bottom-left section is titled 'Thank you for Contacting us' and contains the text: 'Thank you for sending us your inquiries. We will try to answer à your question in the next year or so.' Below this text is a button labeled 'Ok'.

شكل 4-45 الشكل للحالات في هذا التمرين

25. انقر على عنصر التحكم Button الذي قمت بإنشائه وتعين الحدث On Click للعودة الي الحالة Base.

```
currentState = "
```

26. اذهب الي الحالة contact وقم بتغيير الحدث On Click الي لزر التحكم Submit Your Request للذهاب الي الحالة ThankYou.

```
currentState = 'ThankYou'
```

27. قم بحفظ وتشغيل التطبيق، يجب ان تشاهد تغير مظهر التطبيق عند النقر على الازرار المختلفة.

من الجدير بالاهتمام قضاء قليل من الوقت لمشاهدة التعليمات البرمجية المتعلقة بما تم انشائه في نافذة عرض التصميم.

الحالات والتعليمات البرمجية:

في معظم اقسام هذا الكتاب نحاول ان نعرض التصميم والتعليمات البرمجية المستخدم في انشاء التصميم، في رأي ان هذا يمنحك قدرة عالية على التحكم في تطبيقاتك، بلا شك ان استخدام نافذة عرض التصميم في انشاء الحالات هو الخيار الأفضل، لان التعليمات البرمجية يمكن ان تكون شاقة وتحتاج الي مزيد من الجهد خصوصا للمبتدئين، لكن هناك بعض المفاهيم المثيرة للاهتمام لم نشاهدها حتى الان، لنلقي نظرة على التعليمات البرمجية التالية (او التبديل الي نافذة عرض التعليمات البرمجية):

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute">
  <mx:states>
    <mx:State name="contact">
      <mx:AddChild position="lastChild">
        <mx:Panel x="307" y="10" width="250"
height="300" layout="absolute"
title="Send Us a Question">
          <mx:Label x="10" y="10"
text="Enter your name:"/>
          <mx:TextInput x="10" y="36"/>
          <mx:Label x="10" y="66"
text="Enter your e-mail address"/>
          <mx:TextInput x="10" y="92"/>
          <mx:Label x="10" y="122"
text="Type your message here:"/>
          <mx:TextArea x="10" y="138"/>
          <mx:Button x="10" y="210"
label="Submit Your Request" click
= "currentState = 'thankYou'"/>
        </mx:Panel>
      </mx:AddChild>
    </mx:State>
    <mx:State name="thankYou" basedOn="contact">
      <mx:AddChild position="lastChild">
        <mx:Panel x="18" y="263" width="250"
height="200" layout="absolute" title =
"Thank You for Contacting Us">
          <mx:Text x="10" y="10" text="Thank you
for sending us your inquiry. We
will try to answer your question
in the next year or so." Width =
"198"/>
          <mx:Button x="82" y="73" label="OK"
click="currentState = ''"/>
        </mx:Panel>
      </mx:AddChild>
    </mx:State>
  </mx:states>
</mx:Application>
```

```

</mx:states>
  <mx:Panel x="10" y="10" width="250" height="200"
    layout="absolute" title="Enemies of Ed">
    <mx:Text x="10" y="12" text="We would like to hear
      from you. Please click on the link below to
      find out how to contact us." width="210"/>
    <mx:LinkButton x="10" y="74" label="Click Here to
      E-mail Us" color="#0000FF" click =
      "currentState='contact'"/>
  </mx:Panel>
</mx:Application>

```

إذا نظرت الي التعليمات البرمجية بعناية، يبدو ان هناك تناقض في تحدثنا عنه، يتم وضع هيكل الحالة بأكمله داخل بداية ونهاية لوسم يسمى `<mx:states>` كل الحالات التي قمنا بتعريفها في هذا المثال هي حالتين (الحالة Base لا تعد). باختصار يتم وضع كل الحالات في التطبيق داخل وسم يسمى States ، كما يتم وضع تفاصيل كل حالة داخل وسم يسمى State.

بمجرد النقر على عنصر التحكم الذي يستدعي معالج الحدث `currentState`، الوسم او الفئة `AddChild` تستدعي الخاصية `<mx:States>` للعثور على الوسم او الفئة `<mx:state>` المناسبة.

التمرير والحالات :Rollovers and Stats

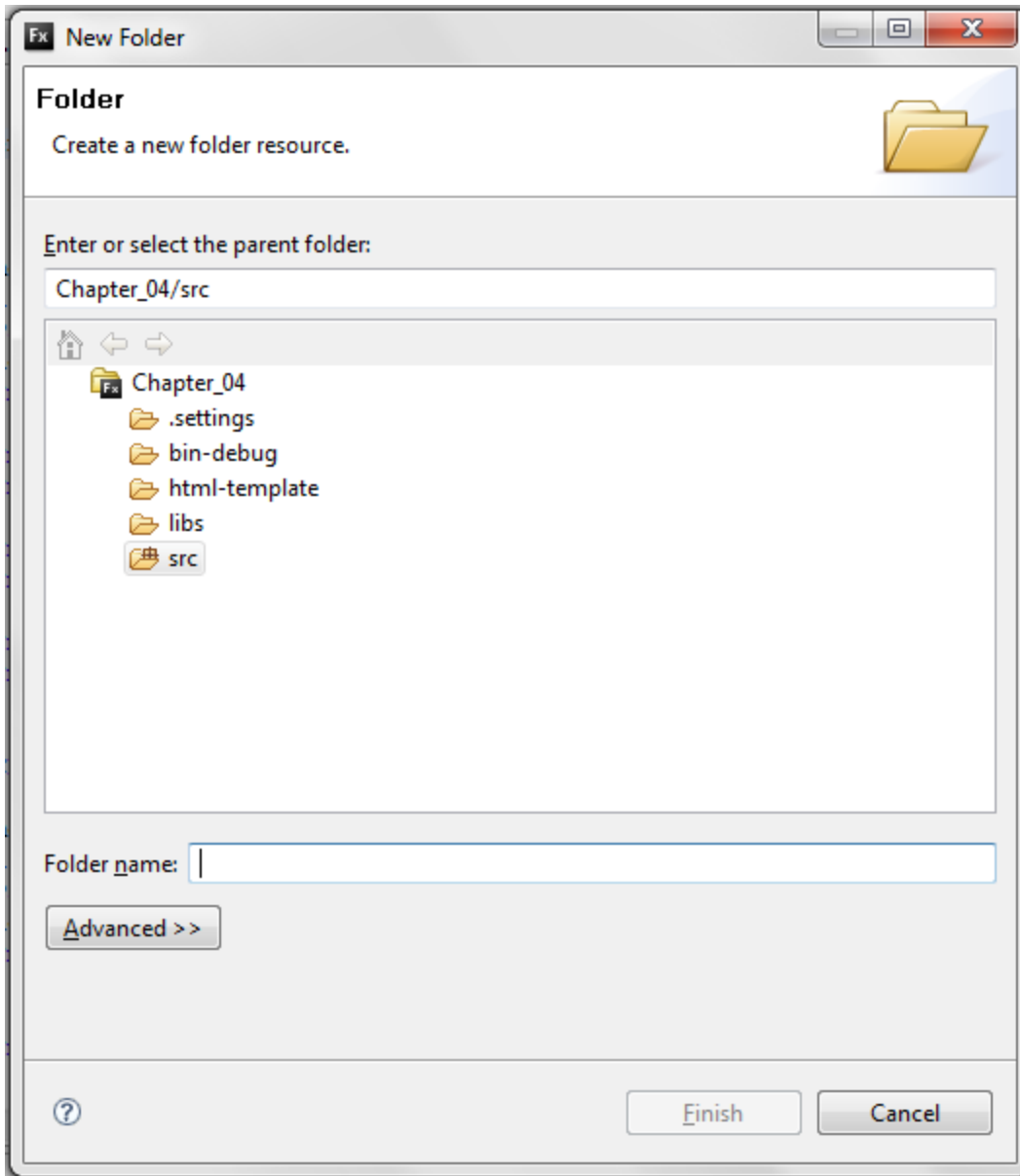
في تصميم صفحات xHTML التقليدية يمكن استخدام JavaScript لإنشاء تأثير التمرير (Rollovers). بمجرد تمرير الفأرة عبرة النص او الصورة، يقوم معالج الحدث في جافا سكريبت بالتقاط الحدث ويصدر تعليمات الي المتصفح بتبديل عنصر باخر.

مرة أخرى نستخدم الحالات لتطبيق حدث التمرير على العنصر، دعنا نحاول ذلك بمثال بسيط، في هذه المرة بدلا من استخدام طريقة عرض التصميم كما كان سابقا نستخدم طريقة عرض التعليمات البرمجية، قبل ان نقوم بذلك نحن بحاجة الي استيراد بعض أصول جاهزة (assets) الي مشروعنا.

استيراد الموارد Resources الجاهزة الي المشروع:

اتبع الخطوات التالية لاستيراد موارد جاهزة الي المشروع الخاص بك:

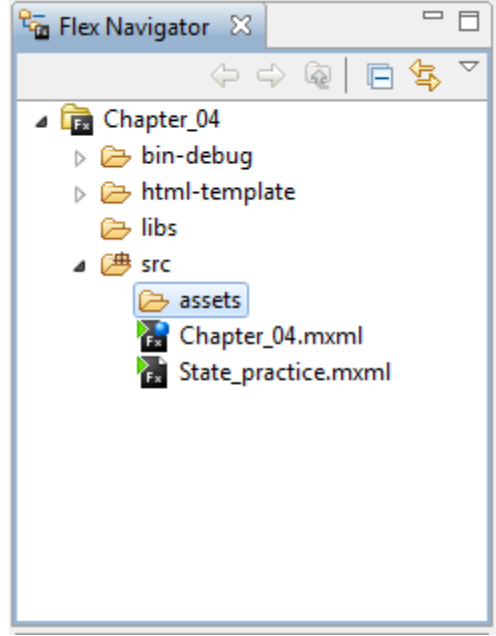
1. اذهب الي نافذة عرض المستكشف Flex Navigator View.
2. انقر بزر الماوس الأيمن على المجلد SRC واختر **Folder < New**، ليتم عرض صندوق حوار كما موضح في الشكل ادناه.



شكل 4 - 46 صندوق حوار انشاء مجلد جديد

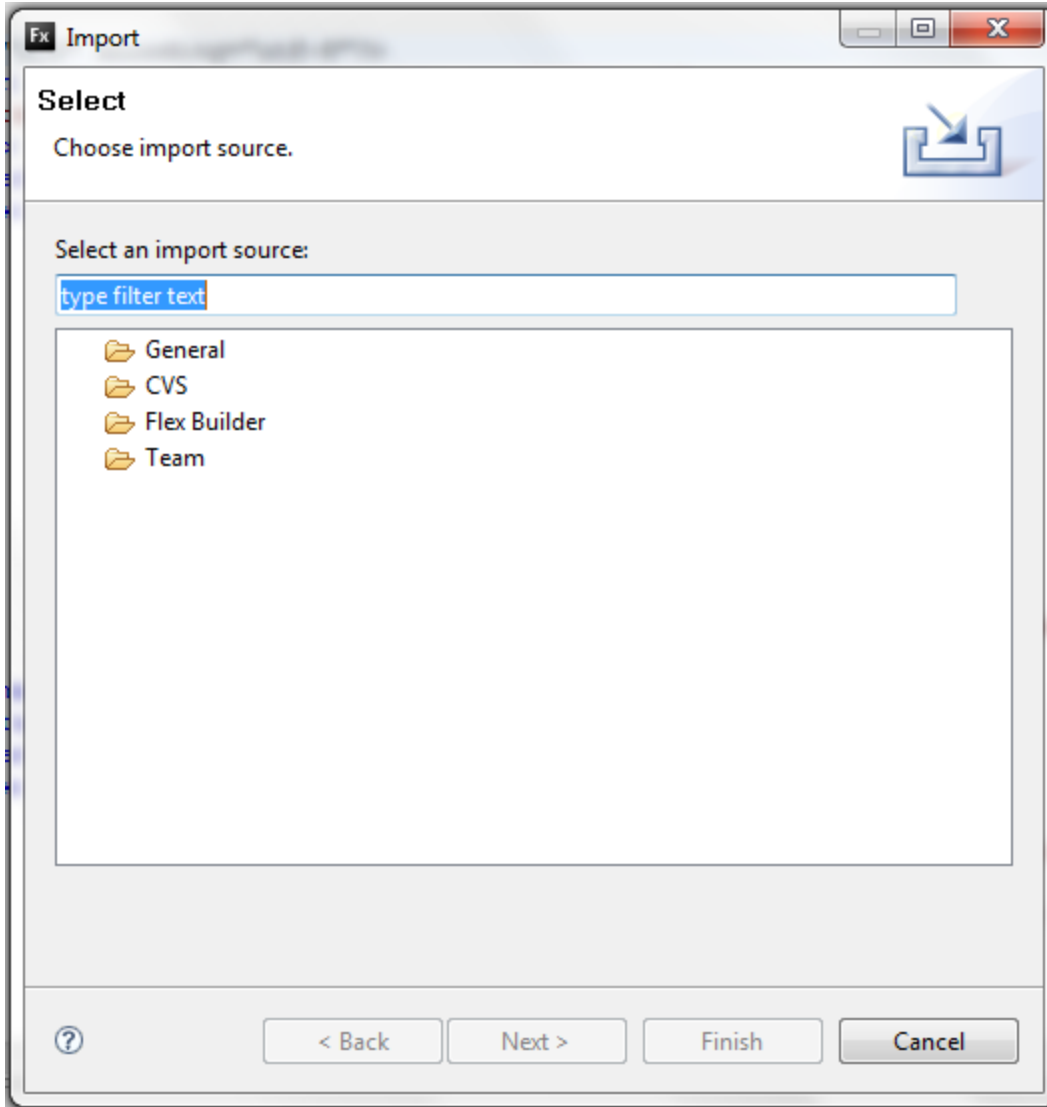
3. سمي المجلد بـ assets وانقر على Finish.

يجب ان تشاهد المجلد الجديد في نافذة عرض المستكشف Flex Navigator كما موضح في الشكل ادناه.



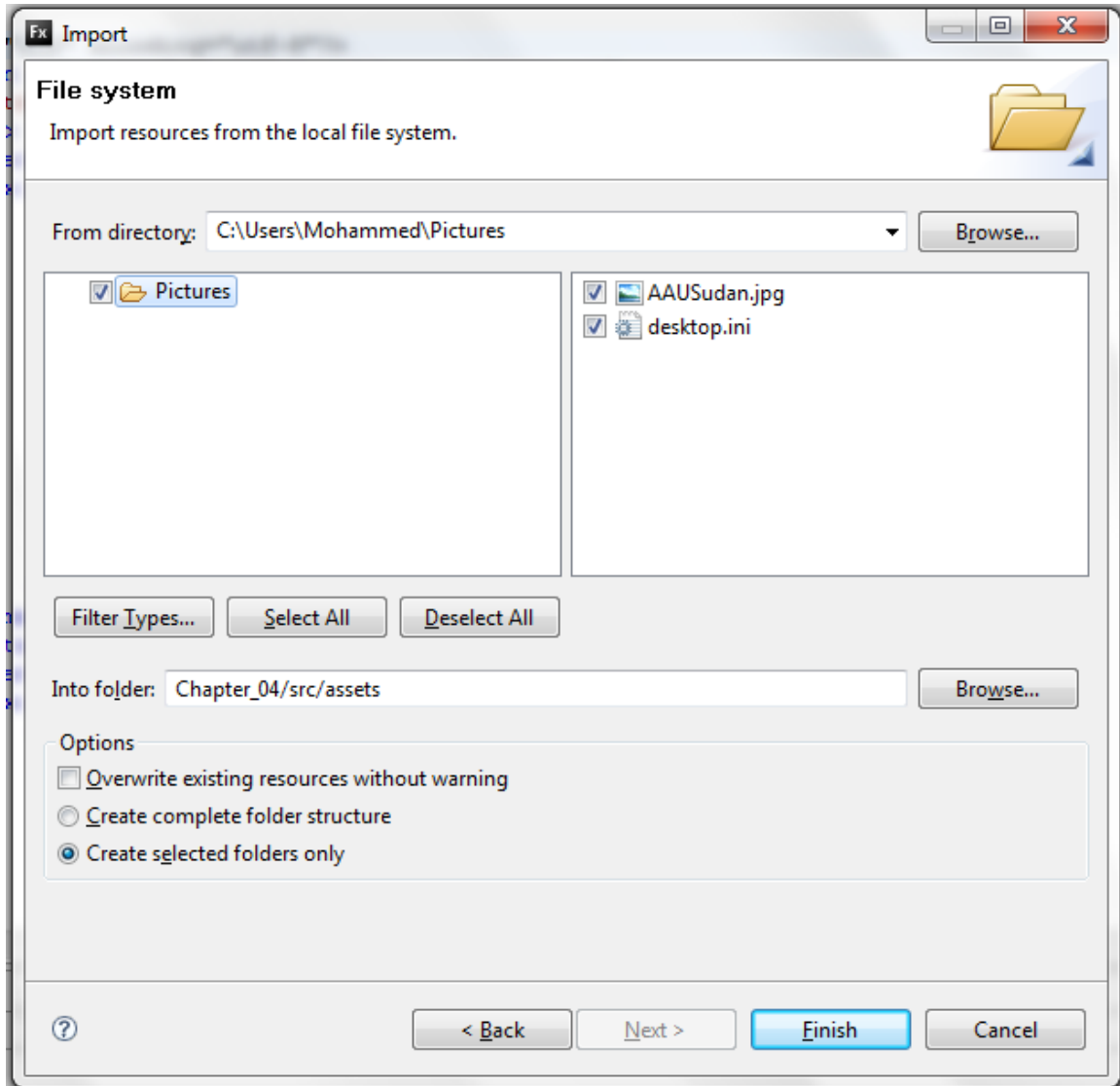
شكل 4 – 47 نافذة عرض المستكشف ويظهر المجلد assets

الخطوة التالية هي استيراد الملفات المطلوبة في هذا المجلد، يمكن ان تختار أي صورة من جهازك
4. انقر بزر الماوس الأيمن على المجلد الجديد assets، في هذه المرة اختر import لظهار صندوق
حوار كما مبين في الشكل التالي.



شكل 4 - 48 صندوق حوار اظهار

5. ضمن الفئة General أخير File System.
6. انقر على Next.
7. انقر على الزر Browse، يمين الحقل Directory Field، وانتقل الي المجلد الذي يحتوي على الملف المراد استيراده الي المشروع، (انظر الي الشكل التالي).



شكل 4 – 49 صندوق حوار ادراج الأصول الموجودة

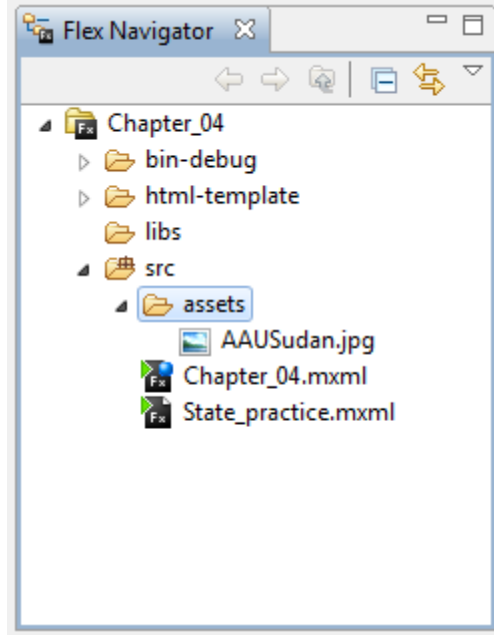
لاحظ ان هناك نافذتين في صندوق الحوار استيراد، يظهر في النافذة اليسرى المجلد ويظهر في النافذة اليمنى الملفات الموجودة داخل هذا المجلد، يمكنك النقر على صندوق الاختيار الموجود شمال المجلد لتحديد كل الملفات داخل هذا المجلد، او النقر على صندوق الاختيار للملفات التي تريدها فقط.

8. اختر الملف AAUSudan.jpg.

9. في الحقل Into folder يجب ان يتم اختيار المجلد src/assets اذا لم يكن كذلك استخدم الزر Browse للانتقال الي هذا المجلد.

10. بعد الانتهاء من تحديد كل شيء انقر على الزر Finish.

ينبغي ان يتم ادراج الملفات في المجلد assets. يمكنك التحقق من ذلك من خلال نافذة مستكشف فيكس.



شكل 4 – 50 نافذة عرض مستكشف فليكس يظهر المجلد الذي تم ادراجه

والان بعد ان تم استيراد الملف نحن مستعدون للعودة لتطبيق مثال تمرير الماوس والحالات.

11. قم بإنشاء تطبيق MXML جديد، و قم بتسمية التطبيق RollOver_practice.

12. قم بضبط تخطيط التطبيق الي Absolute.

13. إذا لزم الامر، قم بالتحويل الي طريقة عرض التعليمات البرمجية.

14. بين بداية ونهاية الوسم Application قم بوضع الوسم Image مع الخصائص الموضحة في

الشكل التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:Image x="180" y="25" source="assets/AAUSudan.jpg" />
</mx:Application>
```

في تطبيقات فلاش، عندما يتم ادراج صور في منصة العمل، يتم تضمين الصورة مع ملف SWF بحيث تصبح الصورة جزء لا يتجزأ من ملف SWF، هذه الحالة ليست هي الحالة المستخدمة مع الوسم Image، الوسم Image يقوم باستدعاء الصورة من الخادم وهذه تشبه كثيرا الطريقة المستخدمة في xHTML، اذا كنت تريد تضمين الصورة في ملف SWF يجب ان تقوم بتعديل الخاصية Source كما موضح في المثال التالي:

```
<mx:Image x="180" y="25" source="@Embed('assets/AAUSudan.jpg')"/>
```

يمكنك التبديل الي طريقة عرض التصميم للتحقق من ان الصورة تم ادراجها بشكل صحيح.
15. الان سنقوم بكتابة التعليمات البرمجية الخاصة بالحالات، لذلك قم بالتبديل الي طريقة عرض التعليمات البرمجية.

في وقت سابق ذكرنا ان كل حالة تستخدم فئة state وكل الفئات state يجب ان تضمن ضمن الفئة .Stats

16. قم بإنشاء وسم States كما موضح في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:states>

  </mx:states>
  <mx:Image x="180" y="25"
    source="@Embed('assets/AAUSudan.jpg')" />
</mx:Application>
```



شكل 4 – 51 طريقة عرض التصميم وتظهر الصورة

يجب استخدام اسم فريد لكل حالة.

17. اصف الحالة التالية واعطي الخاصية name القيمة aau.

```
<?xml version="1.0" encoding="utf-8"?>
```

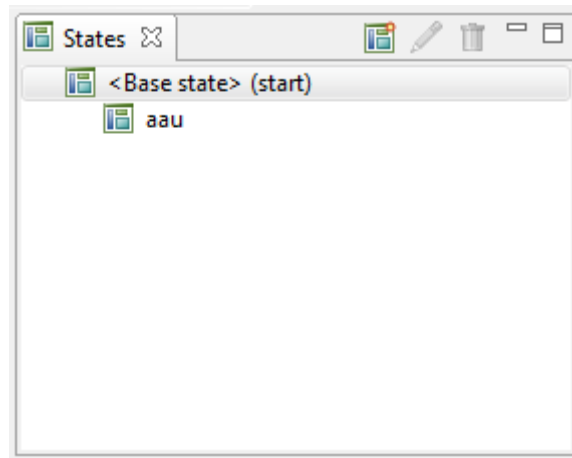
```

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:states>
    <mx:State name="aau">

      </mx:State>
    </mx:states>
    <mx:Image x="180" y="25"
      source="@Embed('assets/AAUSudan.jpg') " />
  </mx:Application>

```

فقط للتحقق قم بالتحول الي طريقة عرض التصميم وقم بإلقاء نظرة الي نافذة عرض الحالات (انظر الي الشكل التالي) يجب ان يتم ادراج الحالة التي انشائها باستخدام الوسم <mx:state> في نافذة عرض الحالات



شكل 4 – 52 الحالة aau التي تم ادراجها

اذهب الي طريقة عرض التعليمات البرمجية، يتم إضافة الحالة الجديدة من خلال انشاء حاوية جديدة باستخدام الفئة AddChild.

18. قم بإضافة الفئة AddChild كما في المثال التالي:

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:states>
    <mx:State name="aau">
      <mx:AddChild position="lastChild">

        </mx:AddChild>
      </mx:State>
    </mx:states>
    <mx:Image x="180" y="25"
      source="@Embed('assets/AAUSudan.jpg') " />

```

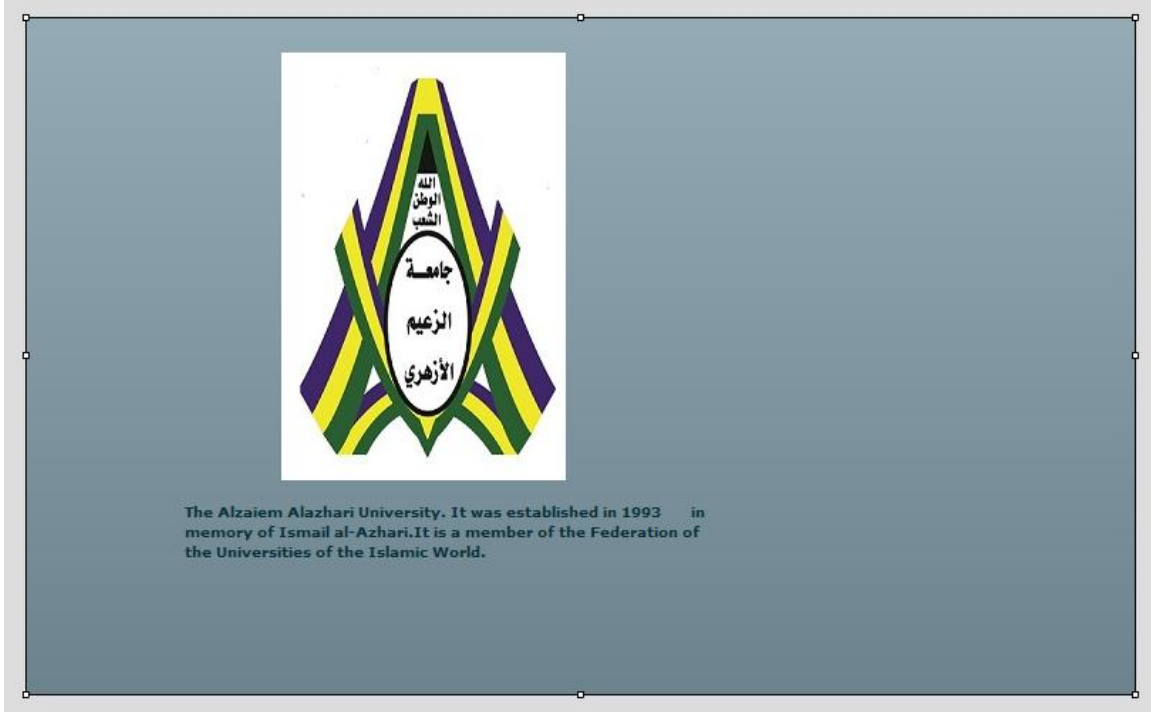
</mx:Application>

داخل الوسم AddChild، نضيف كل المحتويات المطلوبة هذه المحتويات يمكن ان تكون نصوص او صور او حتي حاوية إضافية.

19. في هذا المثال اضف الوسم Text كما موضح في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:states>
    <mx:State name="aau">
      <mx:AddChild position="lastChild">
        <mx:Text width="385" x="110" y="275"
          fontWeight="bold"
          text="The Alzaiem Alazhari University.
          It was established in 1993
          in memory of Ismail al-Azhari.It is a
          member of the Federation of
          the Universities of the Islamic
          World." />
      </mx:AddChild>
    </mx:State>
  </mx:states>
  <mx:Image x="180" y="25"
    source="@Embed('assets/AAUSudan.jpg') " />
</mx:Application>
```

20. قم بالتحويل الي طريقة عرض التصميم وانقر على الحالة aau في نافذة عرض الحالات، ستشاهد مكان النص الخاص بك كما موضح في الشكل التالي:



شكل 4 – 53 مكان النص في الحالة الجديدة

حان الان لنقوم بإضافة بعض الوظائف للتطبيق.

لقد قمنا في وقت سابق باسناد معالج الحدث currentState الي الحدث Click، هنا سنقوم بعمل تغيير بسيط وهو تعيين هذا المعالج الي الحدث Rollover على الصورة.

21. قم باجراء التعديلات التالية على وسم MXML الخاص بالصورة Image:

```
<mx:Image x="180" y="25" source="@Embed('assets/AAUSudan.jpg') "
rollOver="currentState='aau'"/>
```

22. قم بحفظ وتشغيل التطبيق.

عندما تقوم بتمرير الماوس على الصورة يجب ان تظهره المحتويات الخاصة بالحاوية AddChild، لديك الان مشكلة بسيطة وهي عند ابعاد مؤشر الماوس من الصورة تظل محتويات الحاوية AddChild كما هي، يمكنك معالجة هذه الصورة بإضافة حدث ثاني الي عنصر التحكم Image كما موضح في المثال التالي:

```
<mx:Image x="180" y="25" source="@Embed('assets/AAUSudan.jpg') "
rollOver="currentState='aau' " rollOut="currentState=' '"/>
```

23. قم بحفظ وتشغيل التطبيق، يجب ان تلاحظ انه بعد ابعاد مؤشر الماوس من الصورة يختفي النص. دعنا نقوم بإجراء تعديلات بسيطة على التطبيق لنشاهد أساليب البرمجة المثيرة للاهتمام في فليكس.

24. قم بحذف الحدث rollout الذي قمت بإنشائها من عنصر التحكم Image. بعد اغلاق الوسم </mx:State>، سنقوم بإنشاء الوسم SetEventHandler، تسمح الفئة SetEventHandler بتعريف الحدث خارج المكون الذي ينشئ الحدث. هذا مثال بسيط لكن غالبا ما تستخدم هذه الطريقة في الاحداث المعقدة.

الفئة SetEventHandler لها ثلاثة خصائص مهمة هي:

- الخاصية name هي اسم الحدث الذي يتم معالجته. في هذه الحالة سيكون rollout.
- الخاصية target هو الكائن الذي يقوم بارسال الحدث. في هذه الحالة يكون عنصر التحكم Image باستخدام المعرف ID الخاص بالصورة قم بإضافة الخاصية Id الي الوسم Image وليكن aausudan
- الخاصية Handler هو تحدد ما يجب ان يحدث عند وقوع الحدث، في هذه الحالة هي العودة الحالة Base.

25. قم بوضع التعليمات البرمجية التالية بعد اغلاق الوسم States:

```
<mx:SetEventHandler name="rollOut" target="{aausudan}"
handler="currentState=''" />
```

26. قم بتشغيل التطبيق.

الي جانب الحدث SetEventHandler يوجد الحدث SetProperty، بناء جملة الحدث هذه تشبه لحد كبير جملة الحدث SetEventHandler، الخاصية name هنا تحدد الخاصية المتعلقة بالعنصر والمراد تغييرها، وخلافا للحدث SetEventHandler تستخدم الخاصية value لإسناد قيمة للخاصية المراد تغييرها.

في هذا المثال نريد تغيير حجم الصورة بنسبة 50% عندما يتم تنشيط الحالة auu، هذا يتطلب إضافة اثنين وسم من SetProperty واحد للخاصية scaleX والثاني للخاصية scaleY. تستخدم الخصائص scaleX و scaleY لمضاعفة الطول والعرض، لذلك اذا قمت باسناد القيمة 2 الي الخاصية scaleX سوف يتضاعف العرض الحالي بنسبة 2، وكذلك اذا قمت باسناد القيمة 0.5 الي الخاصية scaleY سوف يتم تقليل الارتفاع بمقدار النصف.

27. قم بتجربة ذلك بإضافة التعليمات البرمجية التالية اسفل الوسم SetEventHandler:

```
<mx:SetEventHandler name="rollOut" target="{aausudan}"
handler="currentState=''" />
<mx:SetProperty target="{aausudan}" name="scaleX" value=".5" />
<mx:SetProperty target="{aausudan}" name="scaleY" value=".5" />
```

28. الان قم بتشغيل التطبيق، قم بتمرير الماوس على الصورة، يجب ان يتم تصغير الصورة بنسبة
50%.

الفصل الخامس: الاحداث والمكونات Events and Components

بعد ان تعرفنا على الية عمل تطبيقات فليكس، الان نحن بحاجة للبدء في الحديث عن نماذج التصميم وهذا يعني كيفية تصميم التطبيقات الخاصة بك، وتحتوي هذه التصاميم على تقنيات لجعل التطبيق عبارة عن مجموعة من الوحدات بالإضافة الي تقسيم سير عمل التطبيق.

في هذا الفصل سنتحدث عن الاحداث بشكل مختلف عن الفصول السابقة، بدلا من بناء احداث بسيطة داخل تعليمات MXML البرمجية سنقوم باستخدام Action Script 3.0 كما سنرى، وهذا يوفر الكثير من الإمكانيات بالنسبة لنا.

كما سنتحدث عن كيفية بناء المكونات الخاصة، قد يبدو ان هذا ليس له علاقة بالأحداث لكن في الواقع هذه المكونات مرتبطة بالأحداث، بالإضافة الي ذلك سنتحدث عن Model-View-Controller وهو نمط تصميم يساعد في تخطيط المشاريع.

الاحداث Events:

كما ذكرنا في وقت سابق من هذا الكتاب، الحدث هو أي شيء يحدث يؤدي الي حدوث اجراء، على سبيل المثال يمكن ان يكون النقر على زر الماوس او الضغط على أي مفتاح في لوحة المفاتيح، كما يمكن ان يكون شيء غير مرئي مثل اكمال تحميل التطبيق، تحميل البيانات، اجراء اتصال، والي اخره من الاحداث، تتعامل Action Script مع مئات الاحداث منها المرئية ومنها غير المرئية.

مع ان Action Script 2.0 كانت تدعم كل هذه المهام، الا ان Action Script 3.0 قامت بتبسيطها وجعل التعامل مع هذا المهام أسهل مما كان عليه في Action Script 2.0. سنقوم الان ببناء واجهة رسومية بسيطة، لاختبار عملية التعامل مع الاحداث.

1. قم بإنشاء مشروع جديد في فليكس واختر له أي اسم، انا اسميه Chapter5_Project.

2. قم بجعل تخطيط التطبيق Absolute.

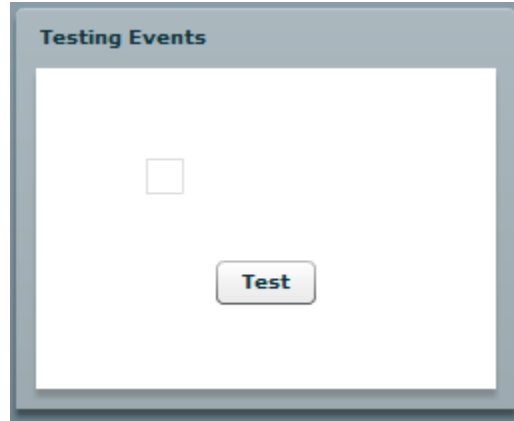
3. قم بإضافة وعاء من النوع Panel مع جعل الخاصية X تساوي 320 والخاصية Y تساوي 130 والخاصية width تساوي 250 والخاصية height تساوي 200، وقم بجعل تخطيط الوعاء

Absolute والعنوان Title يكون Testing Events.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:Panel x="320" y="130" width="250" height="200"
    title="Testing Events" layout="absolute">
```

```
</mx:Panel>
</mx:Application>
```

4. على الوءاء Panel قم بإضافة عنصر التحكم Label مع جعل الخاصية Id تساوي myLabel، وأيضا قم بإضافة عنصر التحكم Button مع وضع الخاصية Id تساوي myButton والخاصية Label تساوي Test، ليس من الضروري تحديد موقع كل عنصر، سنقوم بجعل الخاصية X للعنصر Label تساوي 55 والخاصية Y تساوي 45، ونجعل الخاصية X للعنصر Button تساوي 90 والخاصية Y تساوي 96.



شكل 5 - 1 اعداد واجهة المستخدم الرسومية

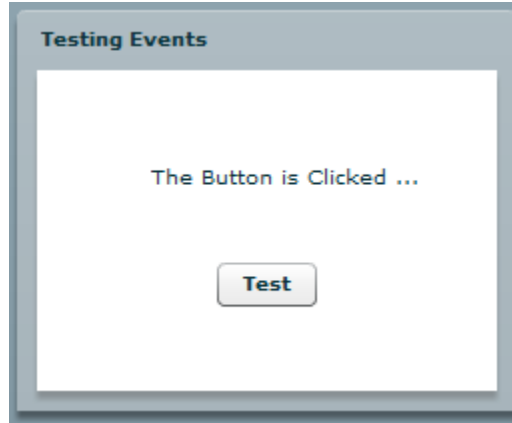
يجب ان يكون الشكل النهائي للتعليمات البرمجية كما يلي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
  <mx:Panel x="320" y="130" width="250" height="200"
    title="Testing Events" layout="absolute">
    <mx:Label x="55" y="45" id="myLabel"/>
    <mx:Button x="90" y="96" id="myButton" label="Test"/>
  </mx:Panel>
</mx:Application>
```

دعونا نقوم بعمل مراجعة سريعة لما سبق، يمكننا ببساطة إعطاء العنصر Button حدث ضمني Inline Event كما في المثال التالي:

```
<mx:Button x="90" y="96" id="myButton" label="Test"
  click="myLabel.text = 'The Button is Clicked'"/>
```

5. قم بتشغيل التطبيق وانقر على الزر Button، يجب ان يكون العنصر Label كما موضح في الشكل 5 - 2.



شكل 5 – 2 التطبيق الاولي

من السهل جدا اظهار النص الخاص بالعنصر Label، مع قليل من التعليمات البرمجية غير تعليمات MXML، هذا الامر يقودنا الي المرونة في التطبيقات، الان دعونا نبدأ التعامل مع الامر بصورة مختلفة قليلا.

6. تحت الوسم Application، قم بإضافة كتلة التعليمات البرمجية التالية لتعريف الدالة fillLabel:

```
<mx:Script>
  <![CDATA[
    private function fillLabel():void
    {
      myLabel.text = "The Button is Clicked!";
    }
  ]]>
</mx:Script>
```

7. قم بتغيير الحدث Click للعنصر Button كما يلي:

```
<mx:Button x="90" y="96" id="myButton" label="Test"
  click="fillLabel()" />
```

هنا بدلا من استدعاء عنصر التحكم Label بصورة مباشرة، قمنا بتمريرها عبر الدالة fillLabel() في كتلة البرنامج النصي Script.

ملاحظة: عند تعريف محدد الوصل الخاص بالدالة private هذا يعني انه يمكن الوصول للدالة من داخل هذا الملف فقط.

8. الان قم بتشغيل التطبيق، يجب ان يعمل التطبيق كما في السابق.

كيف نستطيع جعل الأشياء أكثر مرونة؟ هذا تعقيد! هناك عدد من المصطلحات علينا ذكرها هنا. في الواقع يشار الي الحدث Click على العنصر Button في معظم بيئات البرمجة بمستمع الحدث Event Listener، وهذا يعني ان وظيفته الوحيدة هي الاستماع لهذا الحدث المسند اليه ان يحدث،

وبمجرد حدوث الحدث فإنه يخبر التعليم البرمجية بالقيام بمهامها هذه التعليم البرمجية تسمى معالج الحدث Event Handler.

مثل هذه التعليمات البرمجية تسمح لك ببناء الاحداث بسهولة، ومع ذلك هناك طريقة ثالثة وفعالة جدا لمعالجة الاحداث هي كائن الحدث Event Object.

كائن الحدث The Event Object:

عندما يحدث حدث ما في ActionScript كما في معظم بيئات البرمجة، يتم انشاء كائن يسمى هذا الكائن كائن الحدث، يحتوي هذا الكائن على اثنين من المعلومات المهمة: الهدف target والنوع type. الهدف Target: يحتوي تقريبا على كل المعلومات المتعلقة بمنشئ الحدث، على سبيل المثال يمكن ان يحتوي على الخاصية ID والخاصية X-Position والخاصية Y-Position وما الي ذلك بالنسبة للعنصر Button.

النوع Type: تقوم هذه الخاصية بإرجاع أي نوع من الاحداث تم انشاءه، في مثالنا البسيط هذا سيكون الحدث Click، ولكن في حالات أكثر تعقيدا يمكن ان يكون نوع الحدث مختلف. سنرى الان كيف يعمل كائن الحدث وذلك بعمل بعض التعديلات على التعليمات البرمجية في المثال السابق.

1. قم بتعديل بتعديل حدث العنصر Button كما في المثال التالي:

```
<mx:Button label="Test" id="myButton" x="90" y="96"
click="fillLabel(event)"/>
```

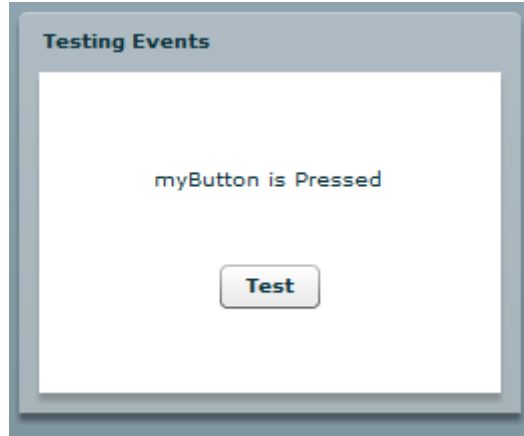
نلاحظ انه عند استدعاء الحدث Click معالج الحدث fillLabel() نقوم بتمرير وسيط، هذه الوسيط يحمل الاسم event، هو في الواقع اسم لهذا الحدث، وهذا الاسم مهم جدا، ولا يمكن استخدام أي اسم اخر غير ذلك.

2. الان، اذهب الي دالة معالج الحدث fillLabel() وقم بتعديل التعليمات البرمجية كما في المثال التالي:

```
private function fillLabel(evt:Event):void
{
    myLabel.text = evt.target.id + " is Pressed";
}
```

نلاحظ اننا حصلنا على كائن الحدث باستخدام مرجع يسمى evt، هنا يمكن ان نطلق عليه أي اسم، هو فقط اسم ليتمكن الدالة من الرجوع اليه.

عندما يتم التقاط كائن الحدث بواسطة الدالة (معالج الحدث)، للدالة صلاحية الوصول الي كافة خصائص منشئ الحدث (تسمى أحيانا مرسل الحدث Event Dispatcher) من خلال الخاصية Target، يمكن للدالة الوصول الي نوع الحدث أيضا.
3. الان، قم بتجربة التطبيق يجب ان يكون الناتج كما موضح في الشكل التالي:



شكل 5 – 3 مخرجات التطبيق

الدالة AddEventListener:

يستهلك مستمع الحدث مساحة كبيرة من الذاكرة، في التطبيقات البسيطة التي قمنا بأنشائها، استهلاك الذاكرة ليس مؤثر، بينما في التطبيقات الكبيرة والمعقدة، نحتاج الي الحد من استخدام الموارد بقدر الإمكان.

توجد الدالة AddEventListener في عدد من ملفات فئات Action Script 3.0، هذه الدالة تسمح لك بإسناد الاحداث الي المكونات، عندما تكون هناك حوجه اليها، سنقوم بعمل مثال بسيط لنفهم ذلك بصورة أفضل:

1. بعد عنصر التحكم Button سنقوم بإضافة عنصر تحكم اخر من النوع Button مع جعل الخاصية Id تساوي myButton2 والخاصية label تساوي Test 2، ونجعل الخاصية X تساوي 90 والخاصية Y تساوي 126.

2. قم بتعديل التعليمات البرمجية في دالة معالج الحدث كما موضح ادناه:

```
private function fillLabel(evt:Event):void
{
    myLabel.text = evt.target.label;
}
```

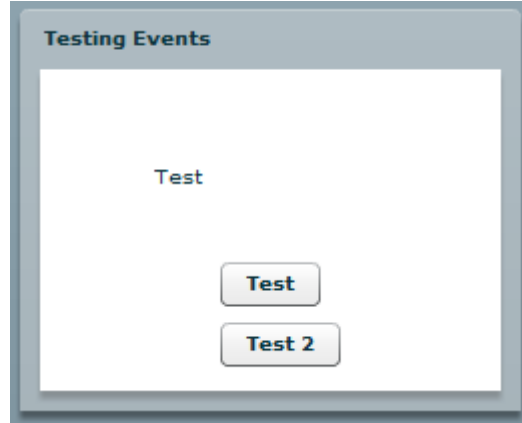
3. بعد التعليمة البرمجية التي قمت بتعديلها، قم بإضافة الدالة `addEventListener`، تأخذ هذه الدالة عدد اثنين معامل، المعامل الأول هو نوع الحدث، المعامل الثاني هو معالج الحدث.
4. قم بكتابة المعامل الأول والذي يمثل نوع الحدث `MouseEvent.CLICK`.
5. قم بإضافة فاصلة، ثم قم بكتابة المعامل الثاني وهو اسم معالج الحدث `fillLabel` يجب ان تكون التعليمات البرمجية في دالة معالج الحدث كما في المثال التالي:

```
private function fillLabel(evt:Event):void
{
    myLabel.text = evt.target.label;
    myButton2.addEventListener(MouseEvent.CLICK , fillLabel
}
```

يجب ان يكون الشكل النهائي للتعليمات البرمجية كما موضح ادناه:

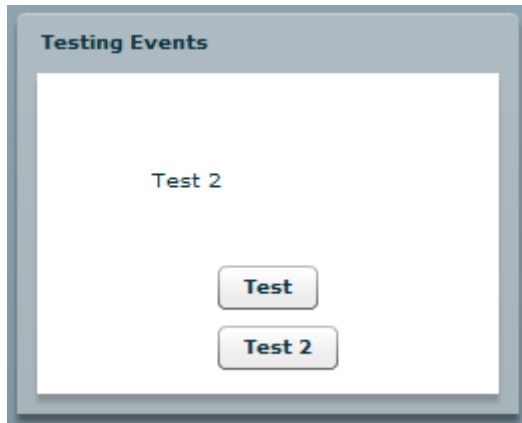
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute">
    <mx:Script>
        <![CDATA[
            private function fillLabel(evt:Event):void
            {
                myLabel.text = evt.target.label;
                myButton2.addEventListener(MouseEvent.CLICK
                    , fillLabel);
            }
        ]]>
    </mx:Script>
    <mx:Panel x="320" y="130" width="250" height="200"
        title="Testing Events" layout="absolute">
        <mx:Label x="55" y="45" id="myLabel"/>
        <mx:Button label="Test" id="myButton" x="90" y="96"
            click="fillLabel(event)"/>
        <mx:Button label="Test 2" id="myButton2" x="90"
            y="126"/>
    </mx:Panel>
</mx:Application>
```

6. قم بتشغيل التطبيق.
7. قم بالنقر على الزر Test 2 أولاً، تلاحظ انه لم يحدث شيء.
8. انقر على الزر Test، ستظهر قيمة الخاصية `label` على عنصر التحكم `Label`، انظر الي الشكل التالي:



الشكل 5 – 4 مخرجات التطبيق عند النقر على الزر Test

9. الان قم بالنقر على الزر Test 2 مرة أخرى، سيكون شكل التطبيق كما موضح في الشكل التالي:



الشكل 5 – 5 مخرجات التطبيق عند النقر على الزر Test 2

بمجرد النقر على الزر Test يتم الوصول الي دالة معالج الحدث، ومن ثم يتم تعيين الحدث في الحقيقة نحن قمنا بتعيين الحدث اثناء تشغيل التطبيق، وهذا يسمى تعيين الحدث وقت التشغيل Runtime، وهذا يعني أيضا بناء على ما يجري يتم تعيين الاحداث حسب الحاجة، قد تتسأل ما أهمية ذلك، هل نريد ان نصل الي انه لا يمكن النقر على الزر A ما لم يتم النقر على الزر B. إذا كنت تفكر كذلك، فانت تفقد نقطة مهمة، أي يمكننا تهيئة الاحداث اثناء وقت التشغيل حسب الحاجة، في هذا المثال لدينا زر وحد يتم من خلاله تهيئة الاحداث لزر اخر هذا مجرد مثال بسيط، لكن دعنا نفكر بصورة مختلفة قليلا من خلال مثال بسيط، لا يمكننا معالجة الحدث الخاص بالزر Button ما لم يقوم المستخدم باختيار صندوق الاختيار Checkbox.

المكونات Components:

حتى هذه اللحظة نقوم بوضع كل المكونات في ملف MXML واحد ونقوم بتنفيذ ذلك الملف، إذا كنت قد قمت من قبل بكتابة تعليمات برمجية باستخدام البرمجة كائنية المنحى OOP، ربما تعلمت انه من الافضل تقسيم الأشياء الي وحدات برمجية تسمى ملف الفئة Class file، وهذا بدوره يساعد في عملية الصيانة وإعادة استخدام التعليمات البرمجية، كل فئة تؤدي وظيفة محددة، وإذا لزم الامر يمكن استدعاء فئات أخرى إذا كنا نريد أداء مهمة إضافية.

حتى الان استخدمنا عدد قليل من الفئات، حيث ان وسوم MXML تعتبر ملفات فئة، وكل وسم في MXML يمكنه الوصول الي الخصائص والوظائف والاحداث المتعلقة بملف الفئة الخاص به.

في هذه القسم نحن نريد انشاء ملفات فئة خاصة، لكن بدلا من كتابة التعليمات البرمجية في Action Script 3.0 سنقوم بكتابتها على في ملف MXML مكافئ يسمى Component.

في البرمجة كائنية المنحى، اتضح ان جميع التطبيقات تقريبا تقع في عدد محدود من أساليب كتابة التعليمات البرمجية تسمى أنماط التصميم Design pattern، عندما يكون التعامل مع هذه الأنماط مألوف بالنسبة لك، فمن السهل اختيار واحدة منها لإنجاز مهمة معينة.

واحدة من أنماط التصميم الأكثر استخداما هو النمط Model-View-Controller (غالبا ما يسمى MVC)، من خلال الأمثلة التالية يمكنك ان ترى عدد من مزايا المكونات من حيث الصيانة وإعادة الاستخدام.

1. قم بحذف كل التعليمات البرمجية بين الوسوم Application في المثال السابق.

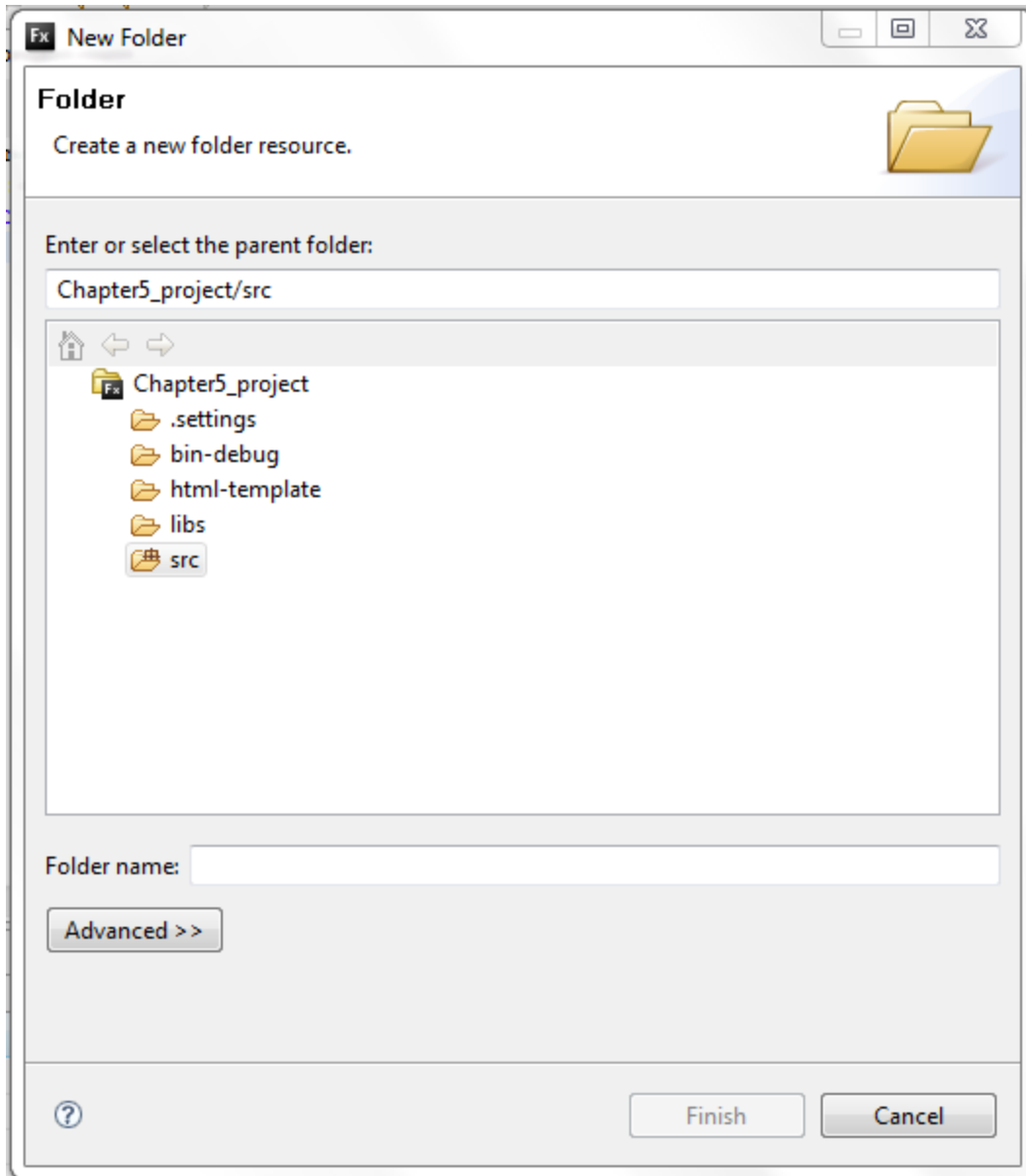
في تصميم تطبيقات MXML يجب ان يكون هناك ملف واحد فقط يحتوي على الوسوم Application يسمى هذا الملف ملف التحكم او الملف الرئيسي او ملف التطبيق.

ليس ضروريا، ولكن من الجيد وضع كل المكونات الخاصة بك في مجلد واحد كما قمنا بذلك عند التعامل مع الصور، هذا يسهل عملية الحصول على الأشياء، في حالة المكونات تسمى هذه المجلدات بالحزم، وكما سنرى لاحقا تتطلب هذه الحزم الي قليل من التعليمات البرمجية والتحضير.

2. في نافذة عرض المتصفح Navigator View، انقر بزر الفأرة الأيمن على المجلد SRC.

3. اختر New ومن ثم Folder.

سيظهر صندوق حوار يسمح لك بتسمية المجلد، كما موضح في الشكل التالي:

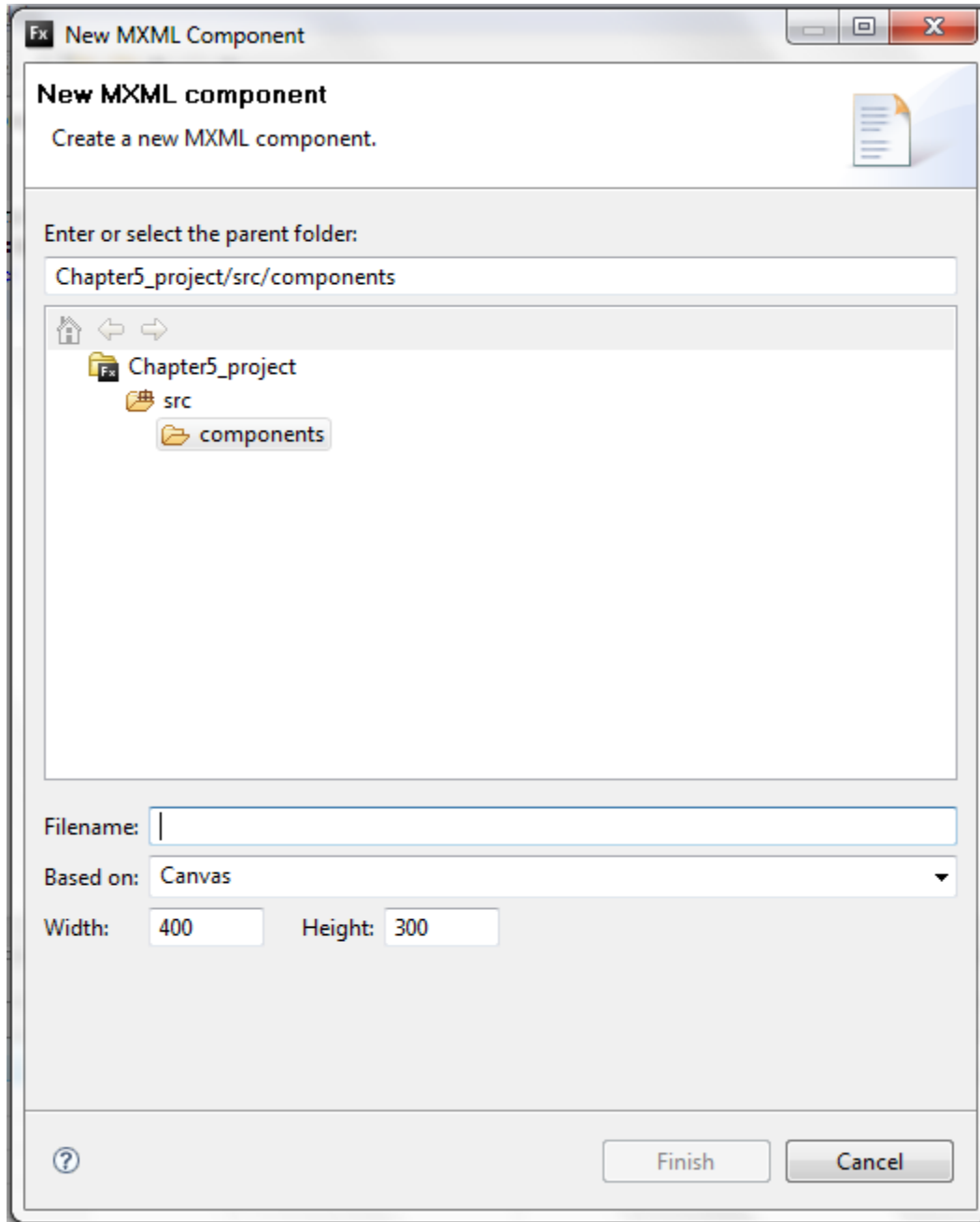


الشكل 5 – 6 صندوق حوار انشاء مجلد جديد

4. في حقل اسم المجلد Folder Name، لقد استخدمنا هنا الاسم components، يمكنك اختيار أي اسم كما تريد.

سنقوم بوضع المكونات داخل هذا المجلد.

5. انقر بزر الفأرة الأيمن على المجلد الجديد components، وأختر New ومن ثم MXML Components، ليظهر صندوق الحوار الموضح في الشكل التالي:



الشكل 5 - 7 صندوق حوار انشاء مكون MXML جديد

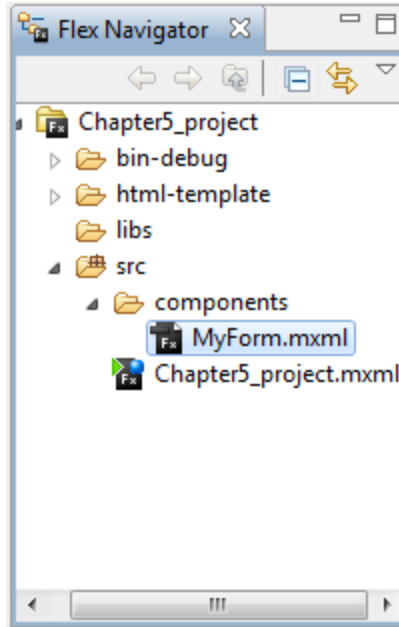
6. في الحقل Filename، ادخل الاسم MyForm ليكون أسم للمكون. إذا نظرنا في الحقل Based on الي القائمة المنسدلة، سترى كل أنواع الالوعية المستخدمة في فليكس.
7. في هذا المثال اختر الالوعية VBox. تلاحظ انه يمكنك ضبط ابعاد المكون العرض width والارتفاع height، لكن الشائع حذف هذه الأرقام في هذا الحقل، هذه يجعل حجم الالوعية يعتمد على حجم المحتويات.

8. قم بحذف قيم الحقل Width والحقل Height، وانقر على الزر Finish. سيتم فتح علامة تبويب جديدة في Flex Builder، لاحظ ان العنصر يستخدم الوسم VBox بدلا من الوسم Application.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">

</mx:VBox>
```

أيضا تلاحظ انه تم عرض المكون في نافذة عرض المستكشف، كما موضح في الشكل التالي:



الشكل 5 – 8 المكون الجديد في نافذة عرض المستكشف

هذا مثال بسيط، سنقوم بإضافة عدد من عناصر التحكم Label الي المكون الجديد، لاحقا سنستخدم امثلة أكثر فعالية مع إمكانية تمرير البيانات.

9. قم بإضافة عناصر التحكم Label الي المكون MyForm كما في الشكل التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Label text="this is a test of our first components"/>
    <mx:Label text="You will quickly see how easy components"/>
</mx:VBox>
```

10. قم بحفظ كل التعديلات، وارجع الي ملف التطبيق الرئيسي.

إضافة المكونات Adding a Components:

هناك عدة طرق يمكن من خلالها إضافة المكون الي التطبيق، سنبدأ بالطريقة الصعبة. كما ذكرنا سابقا يسمى المجلد الذي نضع فيه المكونات بالحزمة، لإدراج مكون في التطبيق يجب ان يكون المترجم قادر على وجود هذا العنصر ومن ثم ادراجه كجزء من ملف SWF الذي تمت ترجمته، لكن كيف يمكن للمترجم العثور على الحزمة؟

في جميع البرامج كائنية المنحى يتم استخدام واحدة من العبارتين: العبارة namespace والعبارة import، مع ان هناك اختلاف بسيط بينهما، الا انها في الأساس تقوم بأداء نفس المهمة، توجه المترجم الي المجلد الصحيح (الحزمة) ومن ثم العثور على المكون او العنصر (ملف الفئة). في فليكس يتم استخدام العبارة import مع Action Script 3.0، وتستخدم namespace مع MXML كخاصية لوسم التطبيق، كما هو مستخدم فعليا مع وسم التطبيق.

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute">
```

مع انه يبدو كإشارة لعنوان موقع انترنت، الا انه في الواقع يشير الي مكتبة فئات ActionScript 3.0 الداخلية.

لاحظ ان الخاصية namespace تظهر بالاسم xmlns وهي تشير الي (XML namespace). بعد النقطتين يمكنك استخدام اسم توكيل بدلا من كتابة المسار في كل مرة تحتاج الي المكون. تستخدم الحروف mx بشكل افتراضي في xmlns، وهذا السبب في ان كل الوسوم تبدأ ب mx.

1. بين وسم البداية والنهاية الخاص بالتطبيق اكتب < وكأنك تريد انشاء وسم جديد وإبداء بكتابة comp كما كنت تفعل، يجب ان تشاهد ان حزمة المكونات الخاصة بك قد انبثقت تلقائيا كما موضح في الشكل التالي:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
3   layout="absolute">
4
5   <comp
6 </mx:Appl
7

```



الشكل 5 – 9 ظهور المكون الجديد على القائمة

2. يمكنك اكمال كتابة اسم المكون او الانتقال الي الأسفل والضغط على المفتاح Enter. بعد اختيار المكون، سيتم إضافة namespace تلقائيا الي وسم التطبيق.

```

<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
layout="absolute" xmlns:components="components.*">

```

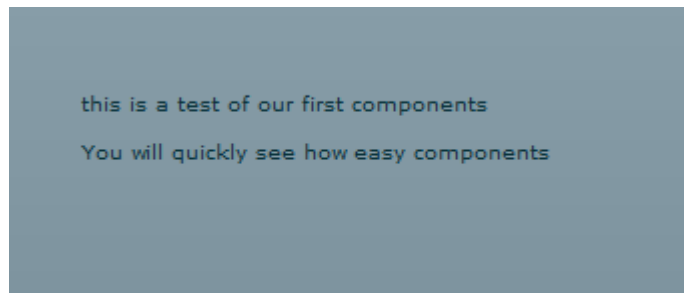
إذا كنت قد قمت بتعيين التنسيق absolute، ستحتاج الي إضافة الخاصية x والخاصية y كأي مكون اخر، يجب ان يكون شكل التعليمات البرمجية كما موضح في الشكل التالي:

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute" xmlns:components="components.*">
  <components:MyForm x="225" y="260"/>
</mx:Application>

```

3. قم بتشغيل التطبيق، يجب ان تكون المخرجات كما موضح في الشكل التالي:



الشكل 5 – 10 المكون الذي تم تضمينه

كما ذكرنا في البداية هذا التمرين ان هذه الطريقة هي الأصعب، سنقوم الان طريقة أسهل.

4. قم بإغلاق أي ملف مفتوح.

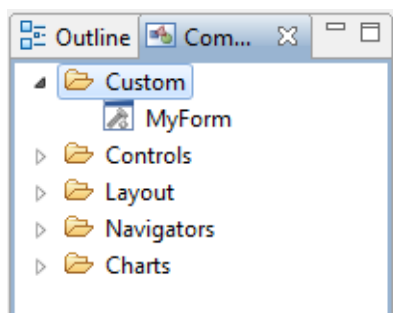
5. قم بإنشاء ملف تطبيق MXML جديد، يمكن ان تستخدم أي اسم، سنستخدم في هذا التمرين

ComponentTest.

6. إذا لزم الامر انتقل الي نافذة عرض التصميم.

إذا نظرت في نافذة عرض المكونات Component View، والتي تقع أسفل الجانب الأيمن من

واجهة Flex Builder، ستشاهد مجلد بالاسم Custom (انظر الي الشكل 5 - 11).



الشكل 5 - 11 Custom في نافذة عرض المكونات

تقوم Flex Builder بإيجاد أي مكون قمنا بإنشائه وإضافته الي المجلد Custom في نافذة عرض المكونات، بمجرد سحب هذا المكون الي منصة التصميم، يتم تلقائيا إضافة الخاصية namespace الي التطبيق.

7. قم بإدراج المكون MyForm الي منصة التصميم، وانتقل الي نافذة عرض التعليمات البرمجية.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" xmlns:ns1="components.*">

    <ns1:MyForm x="217" y="258">
    </ns1:MyForm>

</mx:Application>
```

نلاحظ انه تم تلقائيا إضافة namespace بالاسم ns1، ومن ثم تم استخدامها في انشاء نسخة من

المكون MyForm.

8. قم بتشغيل التطبيق، يجب ان تكون المخرجات كما في التمرين السابق.

هذا مثال بسيط يوضح مفهوم مهم جدا: إعادة استخدام المكونات. تلاحظ اننا قمنا باستخدام نفس المكون في تطبيقات مختلفة، هذا المفهوم بدأ يأخذ أهمية متزايدة.

المكونات والبيانات Components and Data:

1. اضغط بزر الماوس الأيمن على المجلد components في نافذة عرض المتصفح، كما قمنا بذلك سابقا.

2. قم باختيار New > MXML Component.

في هذا المثال نستخدم الوعاء VBox بدون وضع قيم العرض والارتفاع، سنسمي المكون MyForm2، لتمرير البيانات من المكون الي التطبيق والعكس نحتاج الي انشاء متغير لكل خاصية نريد تمريرها.

3. ابدأ التعليمات البرمجية الخاصة بالمكون بإنشاء كتلة تعليمات ActionScript، وقم بتعريف متغيرين مثلا المتغير الأول MyFirstName والمتغير الثاني MyLastName، كلاهما من النوع String.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var myFirstName: String;
      [Bindable]
      public var myLastName: String;
    ]]>
  </mx:Script>
</mx:VBox>
```

تلاحظ: اننا قمنا بتعريف كلا المتغيران كمتغيرات عامة، لأننا نحتاج الي الوصول اليها من خارج المكون، كما ينبغي ان يتم استخدام التعليمة [Bindable] لكلاهما.

4. سنقوم بإضافة اثنين عنصر من النوع Label ليتم ربطهم بالمتغيرين كما موضح في التعليمات البرمجية التالية:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var myFirstName: String;
```

```

        [Bindable]
        public var myLastName: String;
    ]]>
</mx:Script>
<mx:Label id="txtFirstName" text="Your first name is
        {myFirstName}"/>
<mx:Label id="txtLastName" text="Your last name is
        {myLastName}"/>
</mx:VBox>

```

5. سنقوم بحفظ العنصر MyForm2، وإنشاء ملف تطبيق MXML جديد، نسمي التطبيق الجديد .MyNameData

6. سنقوم الان بإضافة متغيرين من النوع String ونسند لكل متغير قيمة ابتدائية، ونجعل محدد الوصول private، كما موضح في المثال التالي:

```

<mx:Script>
    <![CDATA[
        [Bindable]
        private var fName:String = "Mohammed";
        [Bindable]
        private var lName:String = "Mahmoud";
    ]]>
</mx:Script>

```

7. بعد تعليمات ActionScript سنقوم بإضافة العنصر MyForm2 الي التطبيق الجديد، وجعل الخاصية X تساوي 250 والخاصية Y تساوي 125، لا تقوم بإغلاق الوسم نحتاج الي إضافة بعض التعليمات البرمجية هنا، نحتاج الي إضافة الخاصية Id نستخدم هنا الكلمة names.

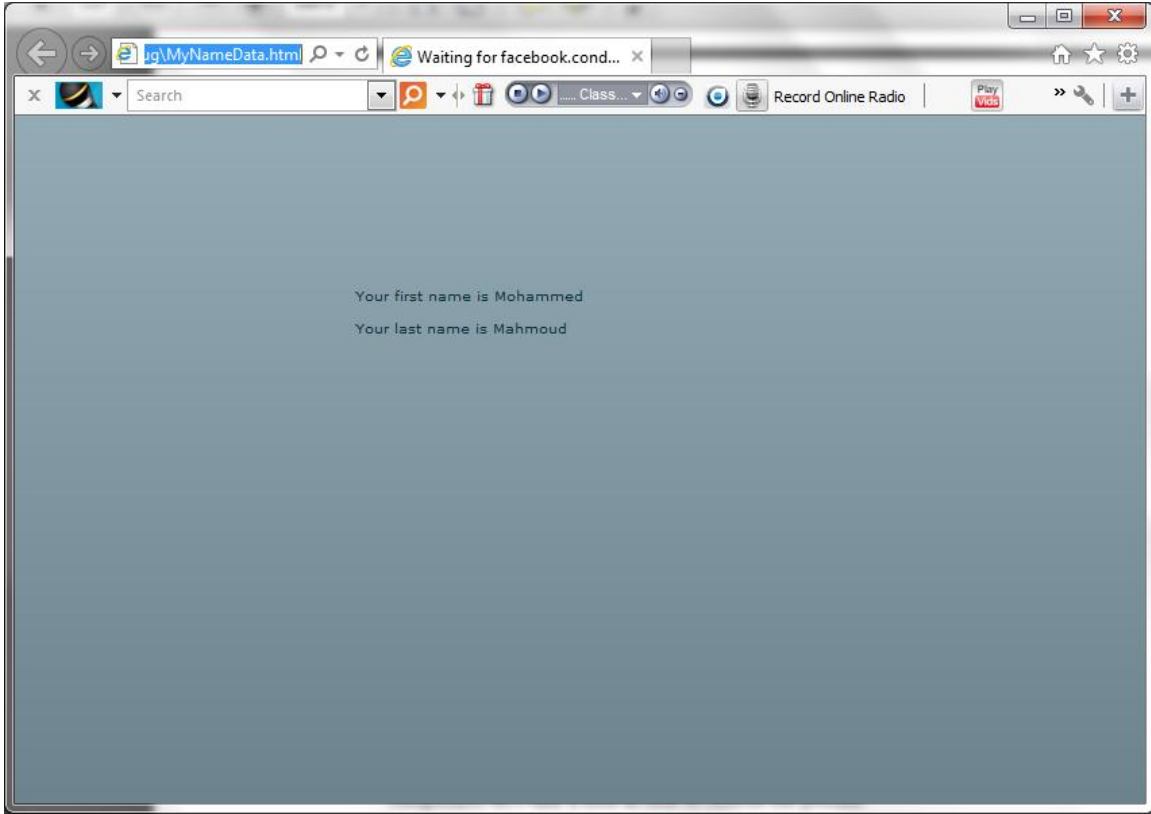
8. سنقوم بإضافة الخاصية myFristName و myLastName وقم بربطهما بالمتغيرين fName و lName، ستكون التعليمات البرمجية كما موضح في المثال التالي:

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" xmlns:components="components.*">
    <mx:Script>
        <![CDATA[
            [Bindable]
            private var fName:String = "Mohammed";
            [Bindable]
            private var lName:String = "Mahmoud";
        ]]>
    </mx:Script>
    <components:MyForm2 x="250" y="125" id="names"
        myFirstName="{fName}" myLastName="{lName}"/>
</mx:Application>

```


9. سنقوم بتنفيذ التطبيق، ستكون المخرجات كما موضح في الشكل التالي:

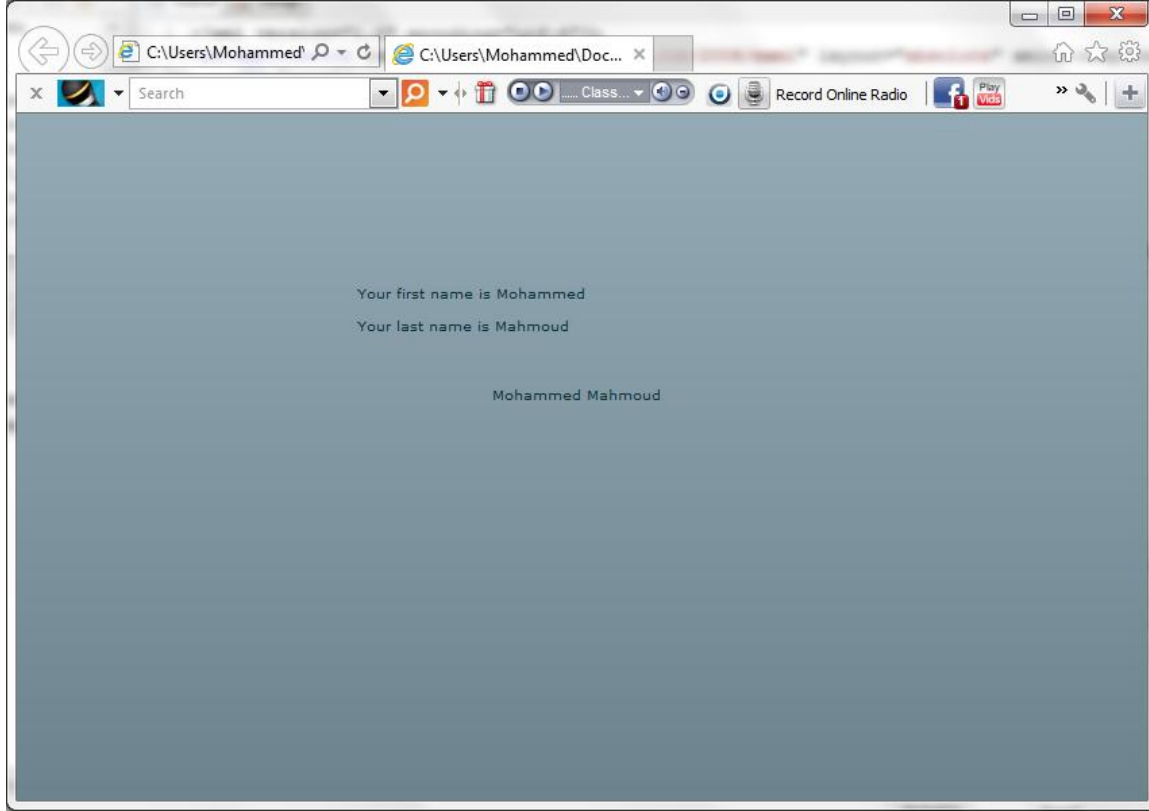


الشكل 5 – 12 تمرير البيانات من التطبيق الي المكونات

تعلمنا في المثال السابق كيفية تمرير البيانات من التطبيق الي العنصر، سنقوم الان بعكس العملية.
10. في ملف التطبيق MyNameData.xml، بعد المكون MyForm2 قم بإضافة العنصر Label، ونجعل الخاصية X تساوي 350 والخاصية Y تساوي 200.
11. سنقوم بربط الخاصية text الخاصة بالعنصر label بخصائص المكون MyForm2 الذي يحمل المعرف names.

```
<mx:Label x="350" y="200" text="{names.myFirstName}  
{names.myLastName}"/>
```

12. قم بإعادة تنفيذ التطبيق، ستكون المخرجات كما موضح في الشكل التالي:



الشكل 5 – 13 تمرير البيانات من المكون الي التطبيق.

الاحداث المخصصة Custom Events:

هناك ثلاثة أجزاء لإنشاء احداث مخصصة:

- 1- الإعلان عن الحدث باستخدام الوسم الوصفي للحدث.
- 2- انشاء الحدث
- 3- ارسال الحدث

الان سنشرح كل خطوة من هذه الخطوات على بشكل مفرد.

الإعلان عن الاحداث المخصصة Declaring a custom Event:

في هذا التمرين، سنقوم بإنشاء تطبيق MXML واطافة اثنين من العناصر، المثير للاهتمام هو في بداية التطبيق ليس هناك علاقة بين ملف التطبيق والملفات الأخرى، سنقوم بربطها مع بعضها البعض لاحقاً، نقوم أولاً بتحديد المكونات التي تؤدي المهام التي نحتاج اليها في انجاز المشروع وجعلها تعمل مع بعضها البعض.

1. قم بإغلاق كل الملفات الموجودة، وقم بإنشاء ملف تطبيق، لهذا التمرين، نجعل الاسم الخاص بملف التطبيق NameMain.xml.
2. تحت وسم التطبيق قم بإضافة كتلة تعليمات Actionscript وقم بإضافة التعليمة [bindable]، ومتغير من النوع String بالاسم sharedNameData وجعل محدد الوصول private ووضع قيمة ابتدائية للمتغير "default Name".
3. تحت المتغير قم بإنشاء دالة من النوع void بالاسم sharedNameHandler، تستقبل هذه الدالة وسيط واحد من النوع event، محدد الوصول لهذه الدالة private. يجب ان يكون شكل التعليمات البرمجية كما موضح في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" xmlns:components="components.*">

    <mx:Script>
        <![CDATA[
            [Bindable]
            private var sharedNameData:String = "default
                Name";

            private function
            sharedNameHandler (evt:Event) :void
            {

            }

        ]]>
    </mx:Script>
</mx:Application>
```

- سنعود الي هذا الملف لاحقا، لكن الان سنقوم بإنشاء المكون المسؤول عن انشاء الحدث.
4. اضغط بزر الماوس الأيمن على المجلد components الذي قمنا بإنشائه سابقا، ونقوم بإنشاء مكون جديد بالاسم NameDispatcher، اعتمادا على الوعاء VBox، ووضع أي قيمة للعرض والارتفاع.
 5. سنقوم بإنشاء كتلة تعليمات ActionScript أسفل وسم الوعاء VBox، وسنقوم بتعريف دالة من النوع void بالاسم clickHandler لا تحمل أي وسائط.
 6. تحت كتلة تعليمات actionScript قم بإضافة العنصر label، واجعل الخاصية text تساوي Name Dispatcher والخاصية fontSize تساوي 16.

7. تحت العنصر Label، سنقوم بإضافة العنصر Button، ونجعل الخاصية label تساوي Click Me ويتم استدعاء الدالة clickHandler التي تم تعريفها مسبقاً في الحدث Click. يجب ان يكون شكل التعليمات البرمجية كما موضح في المثال التالي:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      public function clickHandler():void
      {
      }
    ]]>
  </mx:Script>
  <mx:Label text="Name Dispatcher" fontSize="16"/>
  <mx:Button label="Click ME" click="clickHandler()"/>
</mx:VBox>
```

نحن الان سنقوم ببناء الحدث المخصص الذي سيتيح للمكونات الأخرى ادراك حدوث هذا الحدث، سنقوم بعمل هذا خطوة بخطوة.

إنشاء الحدث :Creating the Event

لبناء الحدث الخاص بنا، يجب علينا اتباع الخطوات التالية:

1. بعد كتلة تعليمات action script وقبل وسم العنصر label، سنقوم بإضافة وسم تعريف البيانات .MetaData

```
<mx:Metadata>
</mx:Metadata>
<mx:Label text="Name Dispatcher" fontSize="16"/>
```

يزود عنصر تعريف البيانات MetaData مترجم فليكس بالمعلومات التي تصف كيفية استخدام مكونات MXML الخاصة في التطبيق، لا تتم ترجمة وسم تعريف البيانات مع التعليمات البرمجية، ولكن يقدم معلومات للتحكم في عملية تنفيذ وترجمة أجزاء التعليمات البرمجية.

لا بد لنا ان نضع في الاعتبار، لا يمكن وضع تعليمات action script او MXML دخل وسم تعريف البيانات، فقط نضع تعليمات خاصة.

داخل وسم تعريف البيانات، نحن نريد ان نخبر المترجم، اننا قمنا بإنشاء حدث مخصص، ومن ثم إعطاء هذا الحدث اسم من عندنا، نقوم بهذا التصريح باستخدام الاقواس المربعة، كما في تعليمة ربط البيانات [Bindable] في تعريف المتغيرات، وتسمى هذه الاقواس المربعة meta tags.

2. قم بإضافة وسم تعريف الحدث كما في المثال التالي:

```
<mx:Metadata>
    [Event(name="nameDataShared")]
</mx:Metadata>
```

لاحظ: قمنا بوضع الخاصية name بين اقواس والاسم الفعلي بين اقواس اقتباس، داخل الدالة clickHandler() نحن بحاجة الي إضافة كائن من الفئة Event والذي يطابق الحدث الذي نقوم بإنشائه، على سبيل المثال سيكون هذا الحدث النقر على الزر Button.

3. سنقوم بإضافة التعليمات التالية الي الدالة clickHandler()

```
<mx:Script>
    <![CDATA[
        public function clickHandler():void
        {
            var myEvent:Event = new Event("nameDataShared");
        }
    ]]>
</mx:Script>
```

الآن سنقوم بالخطوة الأخيرة وهي ارسال الحدث.

ارسال الحدث :Dispatching the Event

المرحلة الأخيرة من انشاء الاحداث المخصصة هي ارسال الحدث، معظم العناصر البصرية في فليكس لها دالة تسمى dispatchEvent()، الحدث الذي نريد ان نرسله سيكون عبارة كائن نسميه myEvent، هذا الكائن يستدعي الحدث المخصص nameDataShared.

1. سنقوم بإضافة الدالة dispatchEvent() بعد الكائن myEvent كما موضح في المثال التالي:

```
public function clickHandler():void
{
    var myEvent:Event = new Event("nameDataShared");
    dispatchEvent(myEvent);
}
```

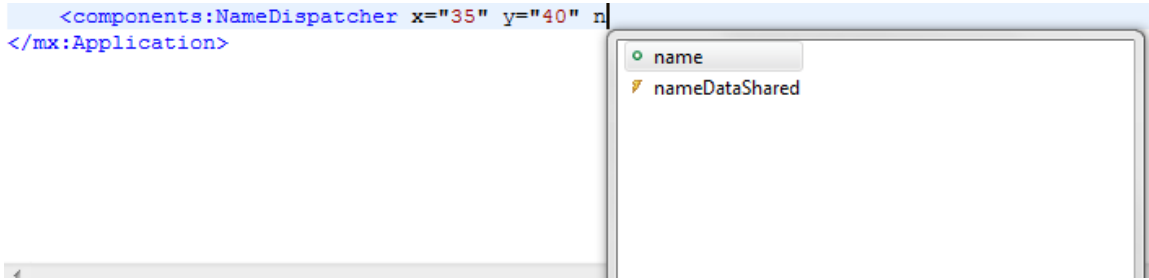
باختصار بمجرد الضغط على الزر Button، تتم الإشارة الي كل المكونات الأخرى المرتبطة انه تم استدعاء الحدث nameDataShared، ما هو نوع الحدث واي كائن قام بإنشاء هذا الحدث وما الي ذلك سيكون مخفي.

2. سننتقل الي ملف التطبيق NameMain.

هنا سنقوم بإنشاء نسخة من المكون الذي قمنا بإنشائه سابقا، بنفس الطريقة التي قمنا باستخدامها في المثال السابق مع اختلاف بسيط.

3. بعد كتلة تعليمات actionscript نقوم بإنشاء المكون nameDispatcher، ونجعل الخاصية X تساوي 35 والخاصية Y تساوي 40.

كما شاهدنا سابقا الاحداث المتوفرة لكل مكون يمكن ادراجها داخل وسم MXML، واستخدما هذا في كثير من الأحيان مع الحدث Click في العنصر Button، الان لدينا حدث مخصص!
4. سنضغط على المفتاح space من لوحة المفاتيح ونكتب الحرف n، نلاحظ ظهور الحدث nameDataShared، ونسميه حدث لأننا نلاحظ علامة الصاعقة التي تشير الي الأحداث تظهر بجانب اسم الحدث.



الشكل 5 – 14 الحدث nameDataShared

5. نختار الحدث nameDataShared، ونكمل كتابة الحدث كما موضح في المثال التالي:

```
<components:NameDispatcher x="35" y="40" n
```

```
nameDataShared="sharedNameDataHandler (event) " />
```

6. داخل الدالة sharedNameDataHandler نقوم بكتابة التعليمات البرمجية التالية:

```
private function sharedNameDataHandler (evt:Event) :void  
{  
    sharedNameData = "Hani Abbass";  
}
```

الخطوة الأخيرة سنقوم بإنشاء مكون بسيط لاستقبال البيانات من المتغير sharedNameData.

7. سنقوم بإنشاء مكون جديد بالاسم ReciveName، باستخدام الوعاء VBox، وعدم وضع قيم للخصائص width و height.

8. سنقوم بإنشاء كتلة تعليمات actionscript واطافة تعليمة ربط البيانات [Bindable] ومتغير من النوع String بالاسم myName

9. بعد كتلة تعليمات actionscript ننشئ عنصر من النوع label، سنجعل الخاصية text تساوي المتغير myName والخاصية fontSize تساوي 16.

10. <?xml version="1.0" encoding="utf-8"?>

```

<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var myName:String;
    ]]>
  </mx:Script>
  <mx:Label text="{myName}" fontSize="16"/>
</mx:VBox>

```

11. الآن سنعود الى ملف التطبيق NameMain، ونقوم بإنشاء نسخة من المكون ReciveName ونجعل الخاصية X تساوي 35 والخاصية Y تساوي 120، وأخيرا سنقوم بربط الخاصية myName بالخاصية sharedNameData في التطبيق.

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="absolute" xmlns:components="components.*">

  <mx:Script>
    <![CDATA[
      [Bindable]
      private var sharedNameData:String = "default
Name" ;

      private function
sharedNameDataHandler (evt:Event) :void
      {
        sharedNameData = "Hani Abbass";
      }
    ]]>
  </mx:Script>
  <components:NameDispatcher x="35" y="40"
nameDataShared="sharedNameDataHandler(event)" />
  <components:ReciveName x="35" y="120"
myName="{sharedNameData}" />
</mx:Application>

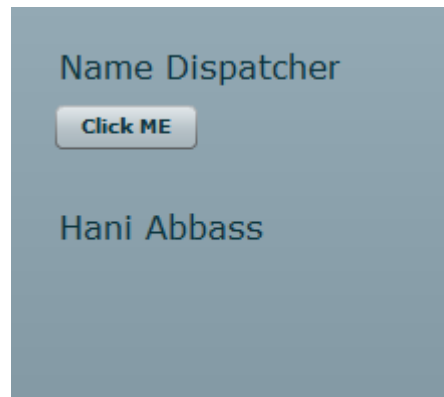
```

12. الآن، سنقوم بتشغيل التطبيق.



الشكل 5 – 15 عند تشغيل التطبيق

13. اضغط على الزر Click ME.



الشكل 5 – 16 عند تنفيذ الحدث المخصص

تمرير البيانات Passing Data:

يمكننا تمرير التعديلات التي حدثت الي المكونات من خلال الخطوات التالية:

1. ننتقل الي المكون NameDispatcher.
2. بعد العنصر Label وقبل العنصر Button سنقوم بإضافة الوعاء HBox.
3. داخل الوعاء HBox سنقوم بإضافة العنصر Label مع جعل الخاصية text تساوي Enter Your Name.
4. تحت العنصر Label سنقوم بإضافة العنصر TextInput، مع جعل الخاصية Id تساوي myNameInput.

يجب ان تكون التعليمات البرمجية الخاصة بالوعاء HBox كما موضح في المثال التالي:

```
<mx : HBox>
```



```

    <mx:Label text="Enter Your Name" />
    <mx:TextInput id="myNameInput" />
</mx:HBox>

```

بما اننا نريد تمرير البيانات، فهذا الامر ليس بالبساطة كالأحداث، لذلك نحن بحاجة الي إيجاد حدث قادر على تمرير البيانات النصية والتعامل معها.

في هذه الحالة نحتاج الي الفئة `TextEvent` الموجودة في حزمة الأحداث `flash.events`، سنقوم الان بعمل بعض التعديلات في التعليمات البرمجية.

5. ننتقل الي وسم وصف البيانات `MetaData`، عندما نريد استخدام فئة حدث غير الفئة `Event`، نحتاج الي الإعلان عن نوع الحدث كما موضح في المثال التالي:

```

<mx:Metadata>
    [Event(name="nameDataShared" , type="flash.events.TextEvent")]
</mx:Metadata>

```

6. الان نذهب الي الدالة `ClickHandler()`، سنقوم بتغيير الكائن `Event` الي `TextEvent`، تأكد من انك قمت بتعديل ذلك في كل الجوانب.

```

public function clickHandler():void
{
    var myEvent:TextEvent = new TextEvent("nameDataShared");
    dispatchEvent(myEvent);
}

```

الفئة `TextEvent` تتيح لنا القدرة على نقل النص من أي حقل، في هذا المكون قمنا بتسمية العنصر `TextInput` بالاسم `myNameInput`، نحن الان بحاجة الي تمرير النص داخل العنصر `TextInput` الي الخاصية `.text`.

7. سنقوم بتمرير النص داخل العنصر `TextInput` الي الخاصية `.text`.

```

public function clickHandler():void
{
    var myEvent:TextEvent = new TextEvent("nameDataShared");
    myEvent.text = myNameInput.text;
    dispatchEvent(myEvent);
}

```

أخيرا، كما اشرنا سابقا، عند استخدام فئة من حزمة مختلفة نحتاج الي ادراج تلك الحزمة، حتى يتعرف المترجم على مكان وجود الحزمة.

8. سنقوم بإدراج الفئة `TextEvent`.

```

<mx:Script>
    <![CDATA[
        import flash.events.TextEvent;

```

```

public function clickHandler():void
{
    var myEvent:TextEvent = new
        TextEvent("nameDataShared");
    myEvent.text = myNameInput.text;
    dispatchEvent(myEvent);
}
]]>
</mx:Script>

```

9. سنقوم بحفظ التعديلات وننتقل الي ملف التطبيق NameMain.mxml سنقوم ببعض التعديلات.

10. سنقوم بتعديل التعليمة sharedNameDataHandler الي TextEvent، وبعد ذلك سنقوم

بإدراج الحزمة flash.events كما في المكون NameDispatcher.

11. سنجعل المتغير sharedNameData يساوي evt.text.

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" xmlns:components="components.*">
    <mx:Script>
        <![CDATA[
            import flash.events.TextEvent;
            [Bindable]
            private var sharedNameData:String = "default Name" ;
            private
            function sharedNameDataHandler(evt:TextEvent):void
            {
                sharedNameData = evt.text;
            }
        ]]>
    </mx:Script>
    <components:NameDispatcher x="35" y="40"
        nameDataShared="sharedNameDataHandler(event)" />
    <components:ReciveName x="35" y="120" myName="{sharedNameData}"/>
</mx:Application>

```

12. الان، سنقوم بتنفيذ التطبيق، سنقوم بإدخال الاسم ونضغط على الزر Click Me، يجب ان يظهر

الاسم على المكون ReciveName.

- The Essential Guide to Flex 3 – June 2, 2008 by [Charles Brown](#) (Author).