**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

**PHẠM QUỐC HUY – TRẦN PHƯƠNG HẢI**

**NGHIÊN CỨU VÀ PHÁT TRIỂN**

**HỆ THỐNG XÂY DỰNG ỨNG DỤNG RIA**

**(RICH INTERNET APPLICATION)**

**TỪ CÁC HỆ THỐNG WEBSITE ĐÃ CÓ:**

**PHÂN HỆ THIẾT KẾ VÀ TỔNG HỢP**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT**

**TP.HCM, 2010**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**PHẠM QUỐC HUY 0612129**

**TRẦN PHƯƠNG HẢI 0612160**

**NGHIÊN CỨU VÀ PHÁT TRIỂN**

**HỆ THỐNG XÂY DỰNG ỨNG DỤNG RIA**

**(RICH INTERNET APPLICATION)**

**TỪ CÁC HỆ THỐNG WEBSITE ĐÃ CÓ:**

**PHÂN HỆ THIẾT KẾ VÀ TỔNG HỢP**

**KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN TIN HỌC**

**GIÁO VIÊN HƯỚNG DẪN**

**TS. TRẦN MINH TRIẾT – NGUYỄN HUY KHÁNH**

**NIÊN KHÓA 2006 – 2010**

|  |
| --- |
| **NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TpHCM, ngày …… tháng …… năm 2010  Giáo viên hướng dẫn |
| **NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  ………………………………………………………………………………  Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.  TpHCM, ngày …… tháng …… năm 2010  Giáo viên phản biện |

LỜI CÁM ƠN

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Tp.HCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài tốt nghiệp này.

Chúng em xin chân thành cảm ơn Thầy Trần Minh Triết, Thầy Nguyễn Huy Khánh và Thầy Nguyễn Đức Huy là những người đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quí báu trong những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến ba, mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Nhóm thực hiện

Phạm Quốc Huy & Trần Phương Hải

ĐỀ CƯƠNG CHI TIẾT

|  |  |
| --- | --- |
| **Tên Đề Tài**: Nghiên cứu và phát triển hệ thống xây dựng ứng dụng RIA (Rich Internet Application) từ các hệ thống website đã có: phân hệ thiết kế và tổng hợp | |
| **Giáo viên hướng dẫn:** TS. Trần Minh Triết – Nguyễn Huy Khánh | |
| **Thời gian thực hiện**: từ ngày 25/01/2010 đến ngày 23/07/2010 | |
| **Sinh viên thực hiện:**  Phạm Quốc Huy (0612129) – Trần Phương Hải (0612160) | |
| **Loại đề tài**: Tìm hiểu công nghệ và xây dựng ứng dụng | |
| **Nội Dung Đề Tài** (mô tả chi tiết nội dung đề tài, yêu cầu, phương pháp thực hiện, kết quả đạt được…):  Mục tiêu đề tài nhằm khảo sát, tìm hiểu các công cụ và kỹ thuật để thiết kế giao diện của ứng dụng web; tìm hiểu các kỹ thuật để phát triển ứng dụng với Silverlight 4; từ đó xây dựng thử nghiệm hệ thống công cụ phần mềm cho phép thiết kế trực quan giao diện của ứng dụng web sử dụng Silverlight và trình diễn ứng dụng web đã được thiết kế.  Đề tài bao gồm các phần sau:   * Khảo sát các hệ thống Mashup (về giao diện) hiện có trên thế giới: **iGoogle**, **Dapper**… Đồng thời, khảo sát, tìm hiểu các công cụ và kỹ thuật thiết kế giao diện của ứng dụng web: **Wix, Moonfruit, SWiSH Max 3, Microsoft Expression Blend 4…** hiện đang có trên thế giới. * Tìm hiểu các kỹ thuật phát triển ứng dụng với Silverlight 4 * Thông qua việc khảo sát hiện trạng của các hệ thống trên cùng với việc tìm hiểu công nghệ Silverlight 4, phân tích các vấn đề cũng như đưa ra giải pháp về kỹ thuật, công nghệ:   + Nêu lên các vấn đề khi xây dựng hệ thống công cụ hỗ trợ thiết kế trực quan.   + Đề ra các giải pháp mang tính kĩ thuật, công nghệ.   + Trên cở sở tìm hiểu và phân tích các vấn đề, nhóm sẽ xây dựng thử nghiệm hệ thống công cụ phần mềm hỗ trợ thiết kế trực quan giao diện của ứng dụng web sử dụng Silverlight. * Trên cơ sở phát triển hệ thống công cụ phần mềm hỗ trợ thiết kế, xây dựng hệ thống để có thể trình diễn lại ứng dụng. | |
| **Kế Hoạch Thực Hiện**:   * 25/01/2010 đến 30/01/2010: Khảo sát hiện trạng chức năng của các hệ thống mashup và công cụ thiết kế web RIA hiện có. * 01/02/2010 đến 28/02/2010: Phân tích và tìm hiểu các kỹ thuật để thiết kế giao diện ứng dụng web. * 01/03/2010 đến 21/03/2010: Tìm hiểu các kỹ thuật để phát triển ứng dụng với Silverlight 4. * 22/03/2010 đến 24/04/2010: Xây dựng công cụ hỗ trợ thiết kế trực quan giao diện của ứng dụng web sử dụng Silverlight. * 25/04/2010 đến 05/05/2010: Xây dựng hệ thống để quản lý sự tương tác giữa các control. * 06/05/2010 đến 21/05/2010: Tiến hành lưu trữ và trình diễn lại ứng dụng web đã được thiết kế. * 22/05/2010 đến 07/06/2010: Hoàn thiện các chức năng chính của hệ thống. * 08/06/2010 đến 30/06/2010: Nâng cấp giao diện và tính tiện dụng của công cụ thiết kế giao diện. * 01/07/2010 đến 23/07/2010: Hoàn thành toàn bộ chương trình dựa trên qui trình và giải pháp đã xây dựng. | |
| **Xác nhận của GVHD** | **Ngày 07 tháng 07 năm 2010 SV Thực hiện** |

MỤC LỤC

[Chương 1 Mở đầu 1](#_Toc267145176)

[1.1. Tổng quan về RIA và Silverlight 1](#_Toc267145177)

[1.1.1. Tổng quan về RIA 1](#_Toc267145178)

[1.1.2. Đặc điểm của Silverlight 3](#_Toc267145179)

[1.2. Ứng dụng phát triển trên nền tảng Web 4](#_Toc267145180)

[1.3. Mục tiêu của đề tài 6](#_Toc267145181)

[1.4. Nội dung của luận văn 7](#_Toc267145182)

[Chương 2 Khảo sát hiện trạng 9](#_Toc267145183)

[2.1. Khảo sát các công cụ Mashup 9](#_Toc267145184)

[2.1.1. Khảo sát xu hướng Mashup hiện nay 9](#_Toc267145185)

[2.1.2. iGoogle 11](#_Toc267145186)

[2.1.3. Dapper 12](#_Toc267145187)

[2.2. Khảo sát các công cụ RIA Web Designer 15](#_Toc267145188)

[2.2.1. Wix 15](#_Toc267145189)

[2.2.2. Moonfruit 17](#_Toc267145190)

[2.2.3. SWiSH Max 19](#_Toc267145191)

[2.2.4. Microsoft Expression Blend 21](#_Toc267145192)

[2.3. Bảng so sánh chức năng 23](#_Toc267145193)

[2.3.1. Bảng so sánh chức năng các công cụ Mashup 23](#_Toc267145194)

[2.3.2. Bảng so sánh chức năng các công cụ RIA Web Designer 23](#_Toc267145195)

[2.4. Kết luận 25](#_Toc267145196)

[Chương 3 Một số vấn đề và giải pháp hỗ trợ thiết kế trực quan 27](#_Toc267145197)

[3.1. Giới thiệu chung 27](#_Toc267145198)

[3.2. Hỗ trợ thiết kế trực quan về vị trí và kích thước 27](#_Toc267145199)

[3.2.1. Hỗ trợ thiết kế vị trí control một cách trực quan 27](#_Toc267145200)

[3.2.2. Xác định control được chọn trên giao diện thiết kế 31](#_Toc267145201)

[3.2.3. Hỗ trợ thiết kế kích thước control một cách trực quan 33](#_Toc267145202)

[3.2.4. Dock các control 35](#_Toc267145203)

[3.3. Lấy, gán và hiển thị động danh sách các thuộc tính 37](#_Toc267145204)

[3.3.1. Lấy và gán giá trị các thuộc tính của control lúc runtime 37](#_Toc267145205)

[3.3.2. Hiển thị động danh sách các thuộc tính ở dạng PropertyGrid 40](#_Toc267145206)

[3.4. Vấn đề control và hiệu ứng 43](#_Toc267145207)

[3.4.1. Tách riêng control và hiệu ứng 43](#_Toc267145208)

[3.4.2. Hiệu ứng BasicEffect 46](#_Toc267145209)

[3.4.3. Thay đổi hiệu ứng cho control dạng BasicListControl 49](#_Toc267145210)

[3.4.4. Gán hiệu ứng cho control lúc runtime 49](#_Toc267145211)

[3.5. List và nguồn dữ liệu 51](#_Toc267145212)

[3.5.1. Giao diện để tùy chỉnh list control (list editor) 51](#_Toc267145213)

[3.5.2. List control và nguồn dữ liệu 52](#_Toc267145214)

[3.6. Phối hợp xử lý giữa các control 53](#_Toc267145215)

[3.6.1. Liên kết sự kiện giữa các control 53](#_Toc267145216)

[3.6.2. Cơ chế khai báo 55](#_Toc267145217)

[3.6.3. Cơ chế lắng nghe và gắng kết sự kiện 56](#_Toc267145218)

[3.6.4. Quản lý những sự kiện được liên kết 57](#_Toc267145219)

[3.6.5. Vấn đề triển khai trên Silverlight 58](#_Toc267145220)

[Chương 4 Một số vấn đề và giải pháp về khả năng mở rộng, sử dụng và triển khai 60](#_Toc267145221)

[4.1. Khả năng mở rộng hệ thống 60](#_Toc267145222)

[4.1.1. Load động một control hoặc hiệu ứng bất kỳ vào chương trình 60](#_Toc267145223)

[4.1.2. Xác định các control và hiệu ứng được chương trình hỗ trợ 61](#_Toc267145224)

[4.1.3. Tạo ra đối tượng control hoặc hiệu ứng 65](#_Toc267145225)

[4.2. Vấn đề ghi nhận và trình diện lại ứng dụng được thiết kế 66](#_Toc267145226)

[4.2.1. Ghi nhận và trình diễn lại những control, hiệu ứng và điều chỉnh của người dùng 66](#_Toc267145227)

[4.2.2. Save và load một đối tượng 67](#_Toc267145228)

[4.2.3. Save và load những tập tin dll cần dùng 71](#_Toc267145229)

[4.2.4. Save và load 1 số thông tin về Canvas chứa control 72](#_Toc267145230)

[4.2.5. Save và load control và hiệu ứng của control 73](#_Toc267145231)

[4.2.6. Save và load các sự kiện đã được liên kết 74](#_Toc267145232)

[4.3. Các vấn đề riêng đối với Silverlight 75](#_Toc267145233)

[4.3.1. Hỗ trợ nội dung HTML trong Silverlight 75](#_Toc267145234)

[4.3.2. Truy xuất tên miền bên ngoài (cross-domain [14]) 77](#_Toc267145235)

[Chương 5 Tổng quan toàn bộ hệ thống 80](#_Toc267145236)

[5.1. Giới thiệu tổng quan 80](#_Toc267145237)

[5.2. Kiến trúc hệ thống 81](#_Toc267145238)

[5.2.1. Tầng Data 82](#_Toc267145239)

[5.2.2. Tầng Data Service 84](#_Toc267145240)

[5.2.3. Tầng Application 85](#_Toc267145241)

[5.3. Kết luận 86](#_Toc267145242)

[Chương 6 Phân hệ thiết kế 87](#_Toc267145243)

[6.1. Giới thiệu 87](#_Toc267145244)

[6.2. Qui trình thiết kế một ứng dụng 89](#_Toc267145245)

[6.3. Kiến trúc 90](#_Toc267145246)

[6.4. Quy trình sử dụng 91](#_Toc267145247)

[6.5. Quy trình mở rộng công cụ 97](#_Toc267145248)

[6.5.1. Mở rộng control 97](#_Toc267145249)

[6.5.2. Mở rộng hiệu ứng 100](#_Toc267145250)

[6.6. Kết luận 101](#_Toc267145251)

[Chương 7 Phân hệ trình diễn 102](#_Toc267145252)

[7.1. Giới thiệu 102](#_Toc267145253)

[7.2. Qui trình tạo lại và quản lý ứng dụng 103](#_Toc267145254)

[7.3. Kiến trúc 104](#_Toc267145255)

[7.4. Tính năng của hệ thống 105](#_Toc267145256)

[7.5. Kết luận 105](#_Toc267145257)

[Chương 8 Kết luận 106](#_Toc267145258)

[8.1. Các kết quả đạt được 106](#_Toc267145259)

[8.2. Hướng phát triển của đề tài 107](#_Toc267145260)

DANH MỤC CÁC HÌNH

[Hình 1‑1 Một ứng dụng xây dựng bằng công nghệ Flash 3](#_Toc267145261)

[Hình 1‑2 Một ứng dụng xây dựng bằng công nghệ Silverlight 3](#_Toc267145262)

[Hình 1‑3 Kiến trúc Silverlight 3](#_Toc267145263)

[Hình 1‑4 Các thành phần của Web 2.0 và ứng dụng trên đó 4](#_Toc267145264)

[Hình 1‑5 Sự tiến hóa của các thế hệ Web 5](#_Toc267145265)

[Hình 2‑1 Giao diện của iGoogle 11](#_Toc267145266)

[Hình 2‑2 Giao diện của Dapper 13](#_Toc267145267)

[Hình 2‑3 Giao diện tạo Flash Widget của Dapper 14](#_Toc267145268)

[Hình 2‑4 Giao diện thiết kế của Wix 15](#_Toc267145269)

[Hình 2‑5 Giao diện SiteMaker của Moonfruit 17](#_Toc267145270)

[Hình 2‑6 Giao diện của phần mềm SWiSH Max 3 19](#_Toc267145271)

[Hình 2‑7 Giao diện của Microsoft Expression Blend 3 21](#_Toc267145272)

[Hình 3‑1 Cách xác định vị trí của control trên Canvas 29](#_Toc267145273)

[Hình 3‑2 Cách xác định vị trí của control trên Grid 29](#_Toc267145274)

[Hình 3‑3 Cách xác định vị trí của control trên StackPanel 30](#_Toc267145275)

[Hình 3‑4 Phân vùng control để xác định khả năng resize 34](#_Toc267145276)

[Hình 3‑5 Giao diện PropertyGrid với các control khác nhau 42](#_Toc267145277)

[Hình 3‑6 Sơ đồ kế thừa của control và hiệu ứng 44](#_Toc267145278)

[Hình 3‑7 EffectableControl 47](#_Toc267145279)

[Hình 3‑8 Quá trình đóng gói control để có thể đặt được hiệu ứng lên control đó 48](#_Toc267145280)

[Hình 3‑9 Ví dụ minh họa về phối hợp xử lý giữa các control [5] 54](#_Toc267145281)

[Hình 3‑10 Các hàm và delegate trong BasicControl liên quan đến việc liên kết sự kiện 56](#_Toc267145282)

[Hình 3‑11 Quá trình lắng nghe và gắng kết sự kiện 57](#_Toc267145283)

[Hình 3‑12 Lớp MDTEventInfo và MDTEventManager 58](#_Toc267145284)

[Hình 4‑1 Cấu trúc thư mục lưu trữ dll và các tập tin liên quan 62](#_Toc267145285)

[Hình 5‑1 Mô hình kiến trúc hệ thống 82](#_Toc267145286)

[Hình 5‑2 Nội dung lưu trữ tầng Data 82](#_Toc267145287)

[Hình 5‑3 Lược đồ cơ sở dữ liệu ở tầng Data 83](#_Toc267145288)

[Hình 5‑4 Các thành phần ở tầng Data Service 84](#_Toc267145289)

[Hình 5‑5 Các thành phần ở tầng Application 85](#_Toc267145290)

[Hình 6‑1 Giao diện chính của công cụ 88](#_Toc267145291)

[Hình 6‑2 Trình đơn ngữ cảnh (context menu) 88](#_Toc267145292)

[Hình 6‑3 Kiến trúc phân hệ thiết kế 90](#_Toc267145293)

[Hình 6‑4 Giao diện ControlTree thể hiện danh sách control được công cụ cung cấp 92](#_Toc267145294)

[Hình 6‑5 Điều chỉnh thuộc tính của control qua Property Grid 92](#_Toc267145295)

[Hình 6‑6 Chọn lựa màu cho control thông qua BrushEditor 93](#_Toc267145296)

[Hình 6‑7 Cài đặt menu item bằng MenuEditor 94](#_Toc267145297)

[Hình 6‑8 Sử dụng RichTextEditor để định dạng dữ liệu để hiển thị cho RichText control 94](#_Toc267145298)

[Hình 6‑9 Gán dữ liệu cho ImageList control thông qua ImageListEditor 95](#_Toc267145299)

[Hình 6‑10 Giao diện danh sách các hiệu ứng được cung cấp cho một loại hiệu ứng 95](#_Toc267145300)

[Hình 6‑11 Giao diện của DataListEditor 96](#_Toc267145301)

[Hình 6‑12 Giao diện EventEditor 97](#_Toc267145302)

[Hình 7‑1 Giao diện chính của ứng dụng Present 102](#_Toc267145303)

[Hình 7‑2 Qui trình tạo lại và quản lý ứng dụng 103](#_Toc267145304)

[Hình 7‑3 Kiến trúc phân hệ trình diễn 104](#_Toc267145305)

DANH MỤC CÁC BẢNG

[Bảng 2‑1 Bảng thống kê và phân loại Mashup Framework dựa theo chức năng [5] 10](#_Toc267145306)

[Bảng 2‑2 Bảng giá các loại tài khoản Premium của Wix 16](#_Toc267145307)

[Bảng 2‑3 Bảng giá các gói dịch vụ của Moonfruit 18](#_Toc267145308)

[Bảng 2‑4 Bảng giá của phần mêm SWiSH Max 3 20](#_Toc267145309)

[Bảng 2‑5 Bảng giá của phần mêm Microsoft Expression Blend 4 22](#_Toc267145310)

[Bảng 2‑6 Bảng so sánh các chức năng của các công cụ đã khảo sát 23](#_Toc267145311)

[Bảng 2‑7 Bảng so sánh các chức năng của các công cụ RIA Designer đã khảo sát 25](#_Toc267145312)

[Bảng 5‑1 Các thành phần chính của hệ thống 81](#_Toc267145313)

[Bảng 5‑2 Dữ liệu ở tầng Data 83](#_Toc267145314)

[Bảng 5‑3 Các thành phần ở tầng Data Service 84](#_Toc267145315)

[Bảng 5‑4 Các thành phần ở tầng Application 85](#_Toc267145316)

[Bảng 6‑1 Các thành phần của phân hệ thiết kế 91](#_Toc267145317)

[Bảng 7‑1 Các thành phần của phân hệ trình diễn 104](#_Toc267145318)

TÓM TẮT KHÓA LUẬN

Ngày nay, với sự phát triển không ngừng của Internet, các ứng dụng web đã dần thay thế một số phần mềm trên desktop, giao diện của các ứng dụng web này gần gũi với người dùng và gần giống như một ứng dụng desktop thông thường. Cùng sự phát triển của Web 2.0, các ứng dụng web được phát triển theo Web 2.0 xuất hiện ngày càng nhiều, một phần trong số đó được xây dựng trên các công nghệ Rich Internet Application (RIA) nên có giao diện khá bống bẩy cùng với những hiệu ứng khá bắt mắt. Tuy nhiên, hiện nay số lượng ứng dụng web sử dụng công nghệ này là không nhiều. Và nếu có sử dụng thì cũng chỉ sử dụng cho một phần nào đó của trang web để hiển thị các thông tin như banner, video, game…. Điều đó làm cho khả năng tương tác và hiệu ứng giữa các thành phần đó cũng chỉ giới hạn trong nội bộ của chúng mà không thể cho toàn bộ trang web. Xuất phát từ thực tế này, nhu cầu tạo nhanh một ứng dụng theo công nghệ RIA dựa trên hệ thống website đã có là không thể thiếu được. Đây cũng chính là mục tiêu trong đề tài nghiên cứu của chúng em.

Nội dung đề tài tập trung vào việc *nghiên cứu và phát triển hệ thống xây dựng ứng dụng RIA từ các hệ thống website đã có: phân hệ thiết kế và tổng hợp.* Hệ thống này sẽ cung cấp một công cụ giúp cho người dùng có thể tạo ra được một ứng dụng RIA một cách nhanh chóng. Với công cụ này, người dùng có thể thiết kế các control, hiệu ứng và điều chỉnh một số thông tin theo nhu cầu của họ. Kết quả của quá trình thiết kế này sẽ được lưu trữ dưới dạng XML. Đồng thời, hệ thống cung cấp chức năng để trình diễn lại ứng dụng đã được thiết kế cho người dùng tại một địa chỉ nhất định. Dựa vào hệ thống, người dùng có thể xây dựng nhanh các ứng dụng web phục vụ cho nhiều lĩnh vực khác nhau như tin tức, bán hàng, web cá nhân…

Nội dung khóa luận bao gồm 8 chương:

**Chương 1:** Mở đầu

**Chương 2:** Khảo sát hiện trạng

**Chương 3:** Các vấn đề và giải pháp trong việc hỗ trợ thiết kế trực quan

**Chương 4:** Các vấn đề và giải pháp trong mở rộng, sử dụng và triển khai

**Chương 5:** Tổng quan toàn bộ hệ thống

**Chương 6:** Phân hệ thiết kế

**Chương 7:** Phân hệ trình diễn

**Chương 8:** Kết luận

# Mở đầu

*✍ Nội dung của chương này trình bày xu hướng phát triển của các thế hệ web hiện nay và nhu cầu của việc xây dựng các hệ thống xây dựng ứng dụng RIA từ các hệ thống website đã có trên Internet đồng thời nêu lên mục tiêu, nội dung và ý nghĩa của đề tài.*

## Tổng quan về RIA và Silverlight

### Tổng quan về RIA

Các ứng dụng web ban đầu được phát triển theo theo mô hình máy chủ (server). Với mô hình này, trách nhiệm của server khá năng nề, server phải đảm trách tất cả các xử lý của trang web, còn client có vai trò chỉ là một trình hiển thị nội dung HTML tĩnh.

Sau đó, khi mô hình ứng dụng client-server xuất hiện, việc phát triển ứng dụng web đã có một bước đáng kể. Với mô hình này, các ứng dụng web không còn một ứng dụng web tĩnh nữa mà đã có thể tương tác được với người dùng. Mô hình này hoạt động theo một nguyên tắc là mọi tương tác liên quan đến dữ liệu đều phải gửi về server, tại server sẽ xử lý yêu cầu được gửi về và trả kết quả xử lý về cho client, khi client đã nhận được kết quả phản hồi sẽ hiển thị kết quả phản hồi vừa nhận được. Chính vì nguyên tắc này mà server phải nhận một lưu xử lý rất lớn, đồng thời làm cho khả năng tương tác với người dùng của ứng dụng trở nên chậm chạp do phải thực hiện việc đồng bộ giữa client và server. Đây là một hạn chế khá rõ nét của mô hình client-server. Để khắc phục hạn chế của mô hình client-server, khuynh hướng chung là chuyển dần các xử lý tương tác về phía client.

Các giải pháp về mặt công nghệ lẫn kỹ thuật như Java Applet, “Plug-in” của Netscape và “ActiveXControl’ của Microsoft lần lượt xuất hiện. Tuy nhiên chúng đều không thể đáp ứng được nhu cầu của người dùng. Và AJAX (*Asynchronous JavaScript And XML*) xuất hiện ]. AJAX có một đặc điểm đó là dựa trên những nền tảng cũ (JavaScript, Document Object Model Level 2, Cascading Style Sheets (CSS)) để xử lý. Chính vì thế, AJAX đã được sử dụng rộng rãi. Với cách thức xử lý bất đồng bộ của AJAX, các ứng dụng web đã có thể tương tác nhanh hơn với người sử dụng do không phải chờ sự phản hồi của server. Tuy nhiên, AJAX vẫn còn hạn chế ở khả năng xử lý đồ họa và multimedia.

Công nghệ RIA đã được *Macromedia* giới thiệu vào tháng 3/2002 ]. Công nghệ RIA đã làm cho các ứng dụng web trở nên mạnh cả về tính năng và giao diện người dùng, mạnh mẽ trong khả năng xử lý đồ họa và multimedia. Hơn nữa, các ứng dụng web được phát triển với công nghệ có giao diện đẹp có khả năng hoạt động độc lập platform, do đó có khả năng triển khai một cách dễ dàng, đây là một điều mà các ứng dụng desktop không thể làm được. Tuy xuất hiện trước hơn cả AJAX (2/2005 ]), nhưng mãi đến gần đây RIA mới được chú ý đến nhiều hơn, trong đó các công nghệ RIA như Flash, Flex và AIR của Adobe, JavaFX của Sun Microsystems, và Silverlight của Microsoft là những công nghệ đang được chú ý nhất hiện nay.

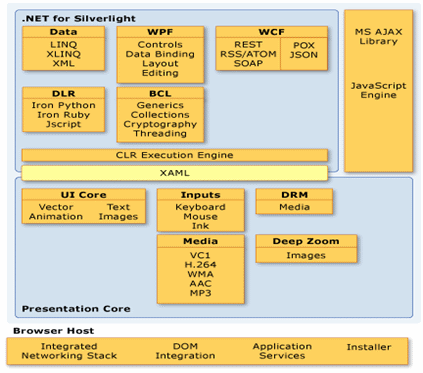
Nguyên nhân của sự phát triển chậm chạp của RIA là do RIA được xây dựng trên một nền tảng mới, do đó cần nhiều thời gian hoàn thiện cũng như cần có thời gian để cộng đồng lập trình viên chấp nhận. Hiện nay, các ứng dụng web sử dụng công nghệ RIA ngày càng nhiều, và các framework RIA cũng liên tục ra các phiên bản mới, gần đây nhất là JavaFX 1.3 và Silverlight 4.0. Những gì mà RIA làm được thực sự ấn tượng.

Đặc biệt với công nghệ Flash và Silverlight, ranh giới giữa ứng dụng web và ứng dụng desktop gần như không còn. Flash và Silverlight đều cung cấp khả năng chạy không cần trình duyệt web, chỉ cần một số hiệu chỉnh nhỏ thì ứng dụng web RIA sẽ chạy trên desktop như ứng dụng desktop thông thường.

|  |  |
| --- | --- |
| www_socnhi_com.png | sirius2_intersoftpt_com.png |
| Hình ‑ Một ứng dụng xây dựng bằng công nghệ Flash  *(Nguồn:* [*http://www.socnhi.com*](http://www.socnhi.com)*)* | Hình ‑ Một ứng dụng xây dựng bằng công nghệ Silverlight  *(Nguồn:* [*http://silverlight.hoyt.net*](http://silverlight.hoyt.net)*)* |

### Đặc điểm của Silverlight

Hình 1‑3 thể hiện những thành phần của kiến trúc Silverlight, cùng với những thành phần và dịch vụ liên quan.



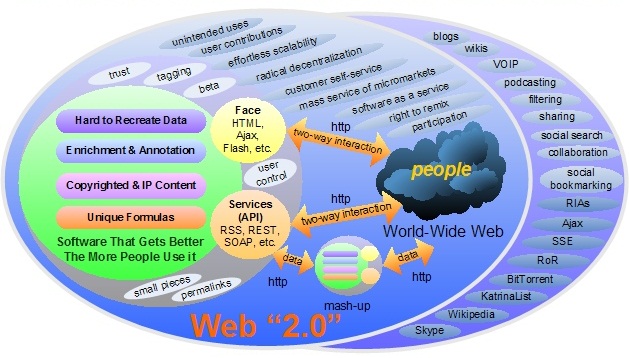
Hình ‑ Kiến trúc Silverlight

*(Nguồn:* <http://msdn.microsoft.com/en-us/library/bb404713%28VS.95%29.aspx>*)*

Phiên bản Silverlight 4 là phiên bản Silverlight mới nhất, ra bản beta vào thàng 11/2009 và bản chính thức ngày 15/04/2010 với rất nhiều cải tiến nội bật:

* + Hỗ trợ mouse right button và mouse wheel.
  + Hỗ trợ WCF RIA services.
  + Hỗ trợ bảo vệ nội dung H.264 và nội dung DRM offline.
  + Hỗ trợ nhiều control mới như RichTextBox, DataGrid…
  + Hỗ trợ Microphone và Camera.
  + Nâng cấp Deep Zoom.
  + Hỗ trợ Managed Extensibility Framework.
  + Hỗ trợ Clipboard và Drag and Drop.
  + Hỗ trợ in ấn.
  + Hỗ trợ truy xuất COM khi chạy out of browser.

## Ứng dụng phát triển trên nền tảng Web

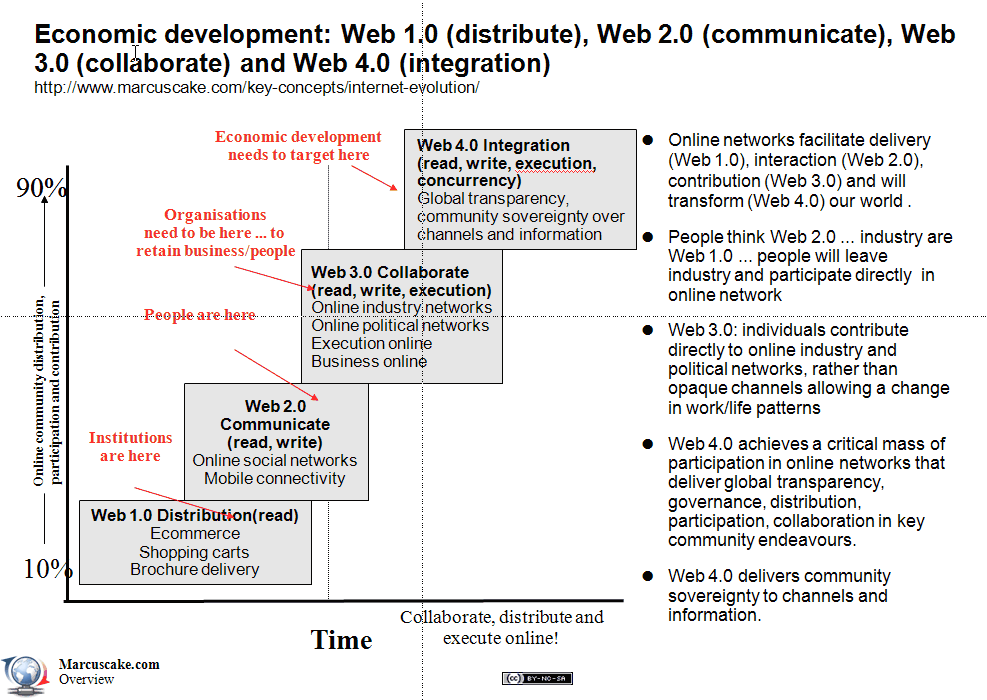


Hình ‑ Các thành phần của Web 2.0 và ứng dụng trên đó

*(Nguồn:* [*http://bragaw.blog.sbc.edu/?p=1597*](http://bragaw.blog.sbc.edu/?p=1597)*)*

Internet ra đời thực sự là một cuộc cách mạng vĩ đại. Internet đã mang lại cho chúng ta một kho kiến thức mà không một thư viện, một bộ bách khoa toàn thư hay một hệ thống thư viện nào khác có thể so sánh được. Internet cũng là môi trường kinh doanh Nhanh - Rẻ - Hiệu quả nhất. Hiện nay, Internet ngày càng phát triển không ngừng cả về cơ sở hạ tầng lẫn những ứng dụng trên nó. Với sự phát triển của Internet và nhu cầu ngày càng đa dạng của người dùng Internet, thế hệ Web 1.0 đã không còn đáp ứng được những nhu cầu đó. Chính vì thế, khái niệm *Web 2.0* đã ra đời. Với trào lưu Web 2.0, các ứng dụng web ngày nay không còn là các trang web tĩnh, hoạt động độc lập nữa, mà nó đã mang tính tương tác cao, tính cộng đồng, hướng về người dùng, thân thiện hơn.

Tuy Web 2.0 đang dần được phổ biến chỉ mới vài năm gần đây, nhưng những khái niệm đầu tiên về thế hệ Web 3.0, Web 4.0 đã xuất hiện ].



Hình ‑ Sự tiến hóa của các thế hệ Web

*(Nguồn:* [*http://www.marcuscake.com/key-concepts/internet-evolution*](http://www.marcuscake.com/key-concepts/internet-evolution)*)*

Nếu Web 2.0 quan tâm đến vấn đề tương tác (interaction) thì Web 3.0 quan tâm đến vấn đề gom nhóm và tổ chức (organization) và Web 4.0 sẽ quan tâm đến vấn đề liên kết lại (integration) để tạo ra một ứng dụng web hoàn thiện.

Với khuynh hướng phát triển của các thế hệ web, các ứng dụng web các trở nên hoàn thiện và dần thay thế các ứng dụng trên desktop. Bởi ứng dụng web có thể sử dụng mọi lúc ở mọi nơi có Internet, và hoàn toàn độc lập với platform. Nắm bắt được xu hướng đó, nhóm chúng em sẽ phát triển hệ thống công cụ hỗ trợ thiết kế giao diện của ứng dụng web trên nền tảng Web và sử dụng Silverlight 4 để làm môi trường phát triển.

## Mục tiêu của đề tài

Mục tiêu đề tài nhằm khảo sát, tìm hiểu các công cụ và kỹ thuật để thiết kế giao diện của ứng dụng web; tìm hiểu các kỹ thuật để phát triển ứng dụng với Silverlight 4; từ đó xây dựng thử nghiệm hệ thống công cụ phần mềm cho phép thiết kế trực quan giao diện của ứng dụng web sử dụng Silverlight và trình diễn ứng dụng web đã được thiết kế.

Nội dung chi tiết đề tài bao bao gồm:

* Khảo sát, tìm hiểu các công cụ và kỹ thuật thiết kế giao diện của ứng dụng web:
* Khảo sát hiện trạng các hệ thống mashup đang có trên thế giới: **iGoogle, Dapper…;** các website và phần mềm để thiết kế web dạng RIA: **Wix, Moonfruit, SWiSH Max 3, Microsoft Expression Blend 4…**.
* Tìm hiểu các kỹ thuật kéo thả trên giao diện.
* Tìm hiểu các kỹ thuật tạo control, tạo hiệu ứng.
* Tìm hiểu các kỹ thuật phát triển ứng dụng với Silverlight 4:
* Tìm hiểu các layout manager được Silverlight 4 hỗ trợ.
* Tìm hiểu kỹ thuật reflection để tạo và thay đổi giá trị của control và hiệu ứng.
* Tìm hiểu cách tạo ra hiệu ứng trong Silverlight.
* Xây dựng thử nghiệm hệ thống công cụ phần mềm cho phép thiết kế trực quan giao diện của ứng dụng web sử dụng Silverlight:
* Xây dựng công cụ để hỗ trợ kéo thả control.
* Tiến hành phân tích và xây dựng một vài control để thử nghiệm.
* Phân tích và xây dựng các hiệu ứng.
* Xây dựng hệ thống quản lý sự tương tác giữa các control.
* Trình diễn ứng dụng đã được thiết kế:
* Xây dựng thành phần save và load ứng dụng đã được thiết kế.

## Nội dung của luận văn

Luận văn bao gồm 8 chương, sau đây là nội dung chính của từng chương:

**Chương 1:** Mở đầu

Nội dung của chương này trình bày xu hướng phát triển của các thế hệ web hiện nay và nhu cầu của việc xây dựng các hệ thống xây dựng ứng dụng RIA từ các hệ thống website đã có trên Internet đồng thời nêu lên mục tiêu, nội dung và ý nghĩa của đề tài.

**Chương 2:** Khảo sát hiện trạng

Nội dung của chương 2 trình bày về việc khảo sát hiện trạng của các công cụ thiết kế web theo công nghệ RIA hiện có trên thế giới.

**Chương 3:** Các vần đề và giải pháp trong việc hỗ trợ thiết kế trực quan

Nội dung chương này sẽ trình bày về các vấn đề khi phát triển công cụ hỗ trợ việc thiết kế trực quan ứng dụng web. Từ đó quyết định các giải pháp để xây dựng công cụ.

**Chương 4:** Các vấn đề và giải pháp trong mở rộng, sử dụng và triển khai

Nội dung chương này sẽ trình bày về các vấn đề cũng như giải pháp cho khả năng mở rộng hệ thống công cụ thiết kế, sử dụng ứng dụng đã được thiết kế và những vấn đề riêng khi triển khai trên Silverlight 4.

**Chương 5:** Tổng quan toàn bộ hệ thống

Nội dung chương này sẽ giới thiệu tính năng của toàn bộ hệ thống đồng thời mô tả kiến trúc của toàn hệ thống. Ứng với mỗi phân hệ chính sẽ có những mô tả tổng quát.

**Chương 6:** Phân hệ thiết kế

Nội dung của chương này trình bày chi tiết qui trình thực hiện, kiến trúc hệ thống và các tính năng của phân hệ thiết kế cộng với các dịch vụ mà hệ thống này cung cấp để hỗ trợ người dùng thiết kế ứng dụng.

**Chương 7:** Phân hệ trình diễn

Nội dung của chương này trình bày chi tiết qui trình thực hiện, kiến trúc hệ thống và các tính năng của phân hệ trình diễn cộng với dịch vụ mà hệ thống này cung cấp để trình diễn lại ứng dụng đã thiết kế.

**Chương 8:** Kết luận

Nội dung của chương này trình bày các kết quả đạt được và hướng phát triển của đề tài.

# Khảo sát hiện trạng

*✍ Nội dung của chương này trình bày về việc khảo sát hiện trạng của các công cụ thiết kế web theo công nghệ RIA hiện có trên thế giới.*

## Khảo sát các công cụ Mashup

### Khảo sát xu hướng Mashup hiện nay

Hiện nay, khái niệm về Mashup vẫn còn chưa được thống nhất và còn nhiều tranh cãi. Theo Wikipedia, *Mashup là một trang web hay một ứng dụng mà sử dụng và kết hợp dữ liệu (data), giao diện (presentation) hoặc xử lý (functionality) từ hai hay nhiều nguồn khác nhau để tạo ra một dịch vụ mới*. Với định nghĩa này có thể thấy các tài nguyên cung cấp cho mashup có thể từ nhiều dạng khác nhau có thể là dữ liệu (RSS Feed…), giao diện (HTML…) hoặc xử lý (Web API…).

Về phân loại, Mashup có thể dựa theo nhiều khía cạnh khác nhau để phân loại Mashup. Với khía cạnh **Loại thông tin tổng hợp Mashup**, Mashup được phân làm ba loại ]:

* Data Mashup (tổng hợp dựa trên dữ liệu): kết hợp dữ liệu từ nhiều nguồn khác nhau để tạo ra một nguồn dữ liệu mới. Nguồn dữ liệu mới này có thể được dùng để cung cấp cho những ứng dụng hoặc xử lý khác.
* Presentation Mashup (tổng hợp dựa trên thành phần thể hiện): dựa trên nội dung, layout được rút trích từ nhiều nguồn trên Web và tích hợp lại thành một giao diện thống nhất.
* Functional Mashup (tổng hợp dựa trên xử lý): Mashup được tạo ra từ sự kết hợp dữ liệu và xử lý từ nhiều khác nhau để tạo ra dịch vụ mới.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Presentation | Data | Functionality | Extraction | Flow | Consumer | Enterprise |
| Apatar |  | x |  |  | x |  | x |
| Data Mashup | x | x |  |  | x |  | x |
| Dapper | x |  |  | x |  | x |  |
| DERI Pipes |  | x |  |  | x | x |  |
| Grazr | x |  |  | x |  | x |  |
| IBM Infosphere Mashup Hub |  | x |  | x |  |  | x |
| Intel Mash Maker | x |  |  |  | x | x |  |
| JackBe Presto |  | x | x |  | x |  | x |
| Microsoft Popfly | x | x |  |  | x | x |  |
| OpenKapow | x |  |  | x |  | x | x |
| Procession |  | x | x |  | x |  | x |
| RssBus | x |  |  | x | x |  | x |
| Serena Mashup Suite |  | x | x |  | x |  | x |
| TIBCO PageBus |  | x |  |  | x |  | x |
| Yahoo Pipes |  | x |  |  | x | x |  |

Bảng ‑ Bảng thống kê và phân loại Mashup Framework dựa theo chức năng ]

là kết quả khảo sát các công cụ Mashup tiêu biểu hiện nay dựa theo chức năng ]. Với kết quả được trình bày ở cho thấy hiện nay đa số các công cụ Mashup chỉ tập trung hỗ trợ việc Mashup về dữ liệu, còn hỗ trợ rất hạn chế việc Mashup về giao diện. Thêm vào đó, các công cụ hỗ trợ Mashup về giao diện còn ở mức hạn chế, thường là dựa trên những Widget được cung cấp sẵn, và đa số những Widget chỉ hoạt động độc lập, không có khả năng tương tác với các Widget khác.

Sau đây, nhóm chúng em sẽ tiến hành khảo sát hai công cụ hỗ trợ Mashup về giao diện là: ***iGoogle*** và ***Dapper***.

### iGoogle

Website: <http://www.google.com/ig>

* Tổng quan hệ thống

iGoogle (trước đây được gọi là *Google Personalized Homepage*, gọi tắt là *Google IG*) là một trong những dịch vụ được cung cấp bởi Google. Đây là trang web cá nhân cho phép tùy biến dựa trên công nghệ AJAX. iGoogle được Goolge đưa ra lần đầu tiên vào tháng 5 năm 2005. Tính năng của iGoogle bao gồm khả năng cho phép thêm các feed từ các trang web có hỗ trợ RSS, và thêm các Google Gadget. iGoogle có số lương Google Gadget tương đối nhiều, vào khoảng 5000 Google Gadget.



Hình ‑ Giao diện của iGoogle

iGoogle có khả năng hổ trợ đa ngôn ngữ (43 ngôn ngữ). Ngoài ra, iGoogle còn cho phép người dùng thay đổi theme của trang web cá nhân. Số lượng theme hiện có của iGoogle khá nhiều, vào khoảng 5000 theme.

Việc sử dụng iGoogle tương đối đơn giản, người dùng chỉ chọn Gadget ưu thích từ danh sách các Gadget có trong iGoogle và cái đặt các thông số cho các Gadget này. Sau khi hoàn thành việc chọn lựa thì Gadget sẽ được hiển thị trên trang chủ của iGoogle.

Google Gadget:

Đây là thành phần được iGoogle sử dụng để tương tác với người sử dụng và sử dụng các Google Gadgets API. Một số tiện ích phát triển cho Google Desktop cũng có thể được sử dụng trong iGoogle. Google Gadgets API là tập API được Google đưa ra để cho những nhà phát triển có thể tạo ra Google Gadget một cách dễ dàng.

Google cũng cho phép tất cả người dùng để tạo ra một Gadget đặc biệt mà không đòi hỏi việc sử dụng các Google Gadgets API.

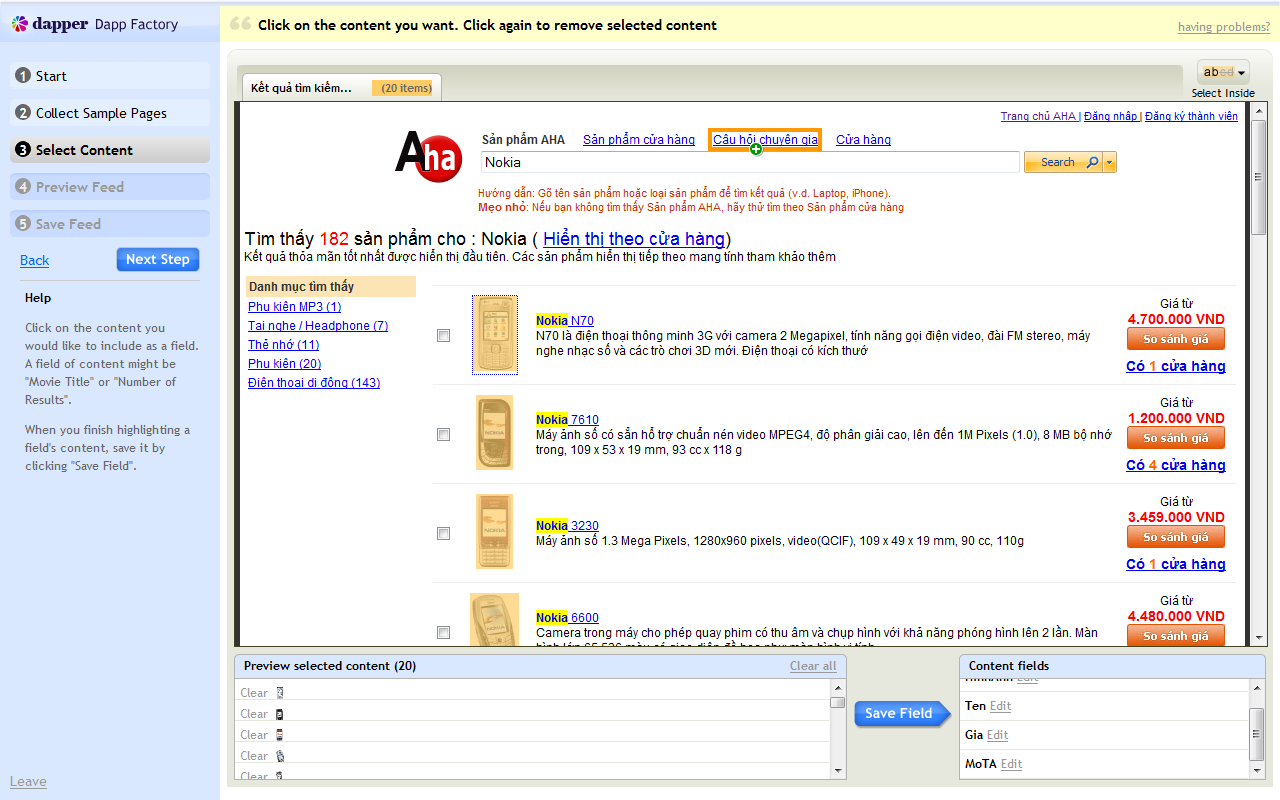
* Ưu khuyết điểm
* Ưu điểm
* Hệ thống miễn phí, người dùng chỉ cần có một tài khoản Gmail.
* Khả năng tùy biến cao, có số lượng Gadget và theme tương đối nhiều.
* Khuyết điểm
* Đa số các Google Gadget chỉ hoạt động độc lập nên không thể phối hợp các Gadget lại với nhau.

### Dapper

Website: <http://open.dapper.net/>

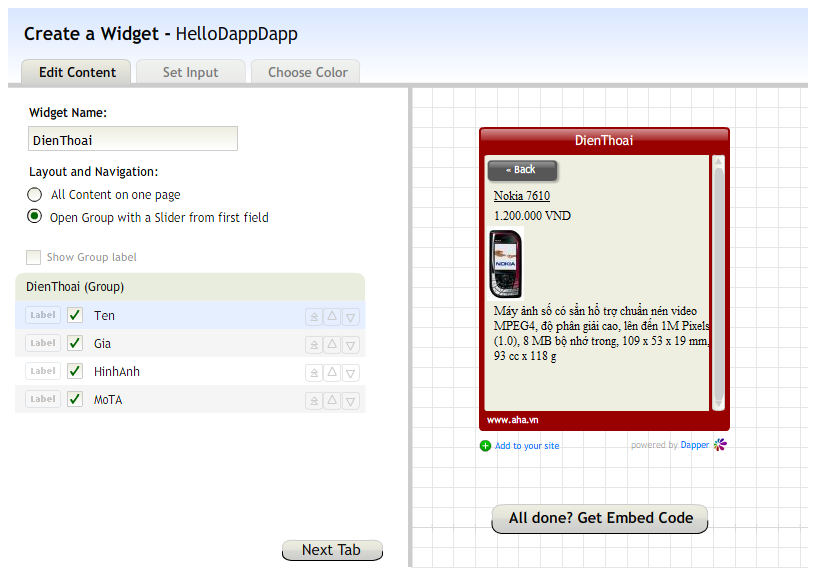
* Tổng quan hệ thống

Dapper được Eran Shir và Jon Aizen thành lập vào năm 2006 để xây dựng một nền tảng chuyển các trang web HTML đơn giản thành các ứng dụng ngữ nghĩa chỉ với một vài thao tác nhấp chuột đơn giản. Bằng việc sử những giải thuật học máy và lý thuyết đồ thị để lập bản đồ cấu trúc cơ bản của trang web, Dapper cho phép bất kỳ người dùng nào đều có thể sử dụng một cách dễ dàng. Dapper được dùng để rút trích dữ liệu từ bất kỳ trang web nào rồi chuyển sang các định dạng khác (XML, RSS…), và tạo ra các ứng dụng.



Hình ‑ Giao diện của Dapper

Sau khi thực hiện các bước để hướng dẫn Dapp Factory tạo ra một Dapp, Dapper cho phép người dùng chọn lựa kiểu định dạng dữ liệu đầu ra (XML, RSS, Google Maps…). Bên cạnh đó, Dapper còn hỗ trợ việc tạo ra Flash Widget. Việc tạo Widget này khá đơn giản tuy nhiên Widget này không có nhiều tùy chọn cho người dùng và chỉ hoạt động độc lập. Dapper còn hỗ trợ người dùng một tập Dapper SDKs để lập trình truy cập đến các Dapp và dữ liệu đầu ra của các Dapp này.



Hình ‑ Giao diện tạo Flash Widget của Dapper

* Ưu khuyết điểm
* Ưu điểm
* Hoàn toàn miễn phí.
* Công cụ có giao diện và cách sử dụng đơn giản và tiện dụng.
* Quá trình tạo Dapp từ các website mà không cần có bất kỳ một thao tác lập trình nào.
* Khuyết điểm
* Các Flash Widget được tạo ra theo mặc định và việc điều chỉnh Widget này rất hạn chế, đồng thời các Widget này chỉ hoạt động đơn lẻ.

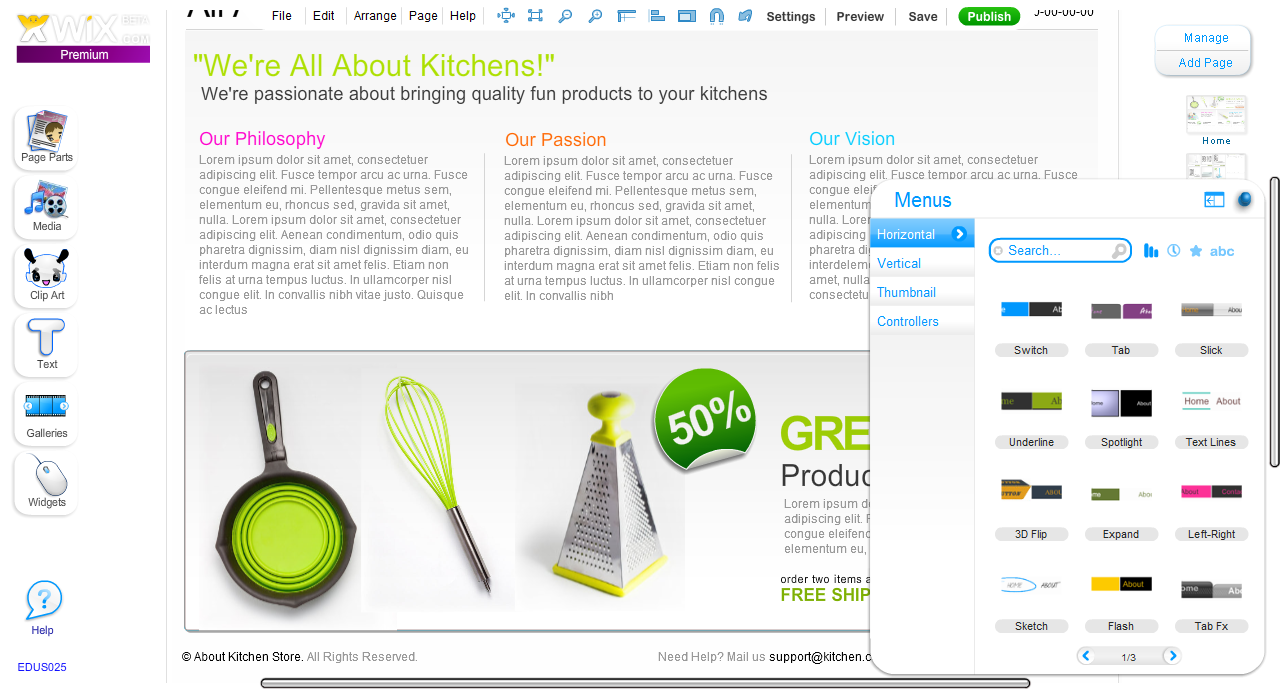
## Khảo sát các công cụ RIA Web Designer

### Wix

Website: <http://www.wix.com/>

* Tổng quan hệ thống

Wix được bắt đầu xây dựng vào năm 2006 bởi Avishai Abrahami, Nadav Abrahami và Giora Kaplan. Wix được ra phiên bản open beta và tháng 6 năm 2008. Wix có một công cụ có giao diện drag drop đơn giản và thân thiện dùng để thiết kế web dạng Flash gọi là *Wix website builder*.



Hình ‑ Giao diện thiết kế của Wix

Các tính năng:

Với công cụ Wix website builder thì người dùng có thể tạo ra những nội dung web (website Flash, widget, banner, sách lật trang…) bằng các thao tác chuột đơn giản và không phải có kiến thúc về lập trình.

Wix website builder cung cấp khá nhiều loại control và hiệu ứng khác nhau. Người dùng sẽ kéo thả các thành phần cần thiết trong một trang vào và xác định Transition, Effect, kích thước, màu sắc cho những thành phần đó. Khi cần liên kết giữa các trang với nhau, người dùng chỉ cần Add Page và thiết kế sau đó tạo ra đối tượng menu thì menu này sẽ tự nhận biết các trang.

Với tài khoản miễn phí thì Wix chỉ hỗ trợ chạy website được tạo ra trên domain của Wix. Nếu muốn chạy ứng dụng trên domain riêng của mình, người dùng cần đăng ký tài khoản premium.

* Giá của tài khoản premium (cập nhật ngày 05/07/2010):

|  |  |
| --- | --- |
| **Loại tài khoản** | **Giá (trong 1 tháng)** |
| My Domain | $4.95 |
| Combo | $9.90 |
| Unlimited | $14.90 |
| eCommerce | $15.90 (6 tháng đầu)  $19.90 (6 tháng sau) |

Bảng ‑ Bảng giá các loại tài khoản Premium của Wix

* Ưu khuyết điểm

Sau đây là những ưu khuyết điểm mà nhóm chúng em nhận xét khi sử dụng công cụ này:

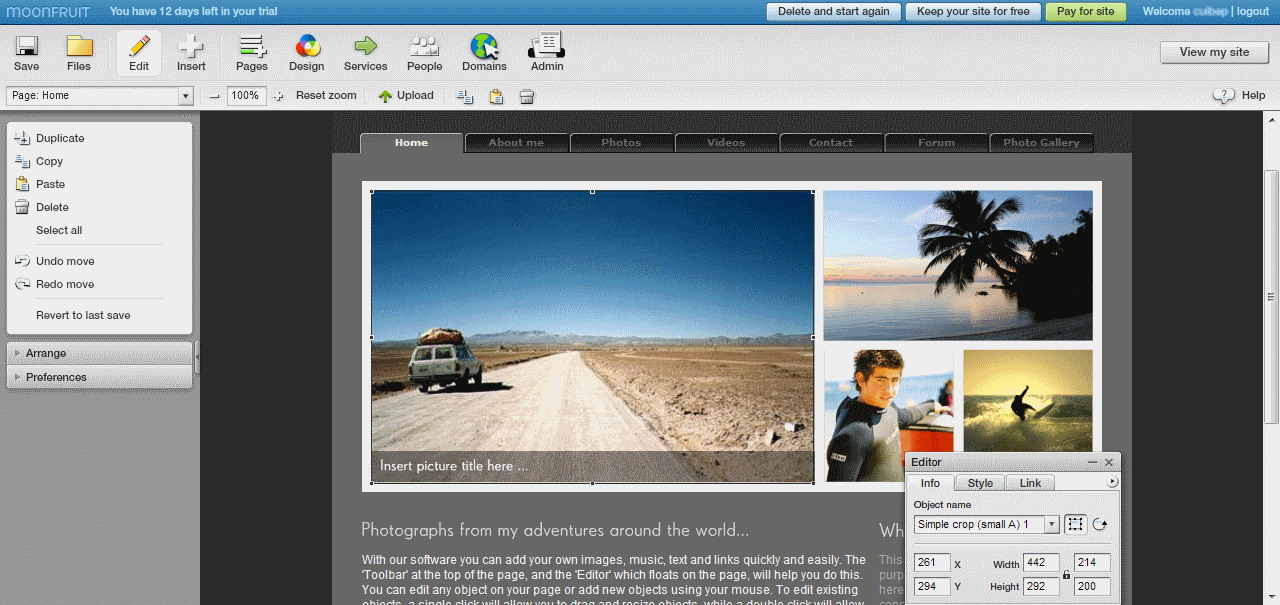
* Ưu điểm
* Công cụ có giao diện và cách sử dụng đơn giản và tiện dụng.
* Wix hỗ trợ sẵn khá nhiều template, control và hiệu ứng.
* Giá của tài khoản tương đối rẻ.
* Khuyết điểm
* Dữ liệu dùng để hiển thị là dữ liệu tĩnh. Do đó ứng dụng tạo ra không thể liên kết dữ liệu động. Ví dụ nếu muốn thêm vào một sản phẩm trong danh sách sản phẩm thì vào công cụ thiết kế để thêm.
* Khả năng tuy biến việc phối hợp hoạt động của các thành phần hạn chế. Việc phối hợp hoạt động do Wix website builder tự xác định.

### Moonfruit

Website: <http://www.moonfruit.com/>

* Tổng quan hệ thống phần mềm

Moonfruit là công ty cung cấp dịch vụ hosting web ở Anh. Moonfruit được thành lập vào năm 1999 và website của moonfruit được xây dựng vào tháng 1 năm 2000.



Hình ‑ Giao diện SiteMaker của Moonfruit

Các tính năng:

Moonfruit cung cấp một công cụ xây dựng trang web bằng flash gọi là *SiteMaker*, nhằm hỗ trợ cho những người không có kiến thức về kỹ thuật vẫn có thể tạo một trang web đơn giản.

Hiện nay, công cụ này đang ở phiên bản thứ tư, trong đó có một phiên bản bằng tiếng Pháp và tương lai sẽ hỗ trợ nhiều ngôn ngữ hơn.

SiteMaker cho phép drag drop và điểu chỉnh đối tượng bằng chuột, cho phép tạo hiệu ứng, điều chỉnh màu sắc và độ trong suốt của đối tương trên trang web. Để thêm một trang, chọn chức năng Add Page từ trên menu của SiteMaker. Moonfruit cung cấp sẵn các template page, với template page đã chứa sẵn các control hoặc widget liên quan đến loại template page đó. Người dùng có thể chọn lựa trang nào sẽ được hiển thị trên menu của trang web bằng chức năng Manage Page.

Đặc biệt, Moonfruit còn cung cấp sẵn template page dạng forum trong đó có sẵn chức năng đăng ký, đăng nhập. Đồng thời cũng cung cấp sẵn hệ thống quản lý user cho website được tạo và cũng liên kết tới các mạng xã hội (Facebook và Twitter).

* Giá của các gói dịch vu của Moonfruit (cập nhật ngày 05/07/2010):

|  |  |  |
| --- | --- | --- |
| **Tên gói dịch vụ** | **Giá (1 tháng)** | **Giá (1 năm)** |
| Lite | $4.49 | $33.99 |
| Standard | $8.99 | $67.99 |
| Plus | $13.99 | $101.99 |
| Max | $23.99 | $186.99 |

Bảng ‑ Bảng giá các gói dịch vụ của Moonfruit

* Ưu khuyết điểm

Sau đây là những ưu khuyết điểm mà nhóm chúng em nhận xét khi sử dụng công cụ này:

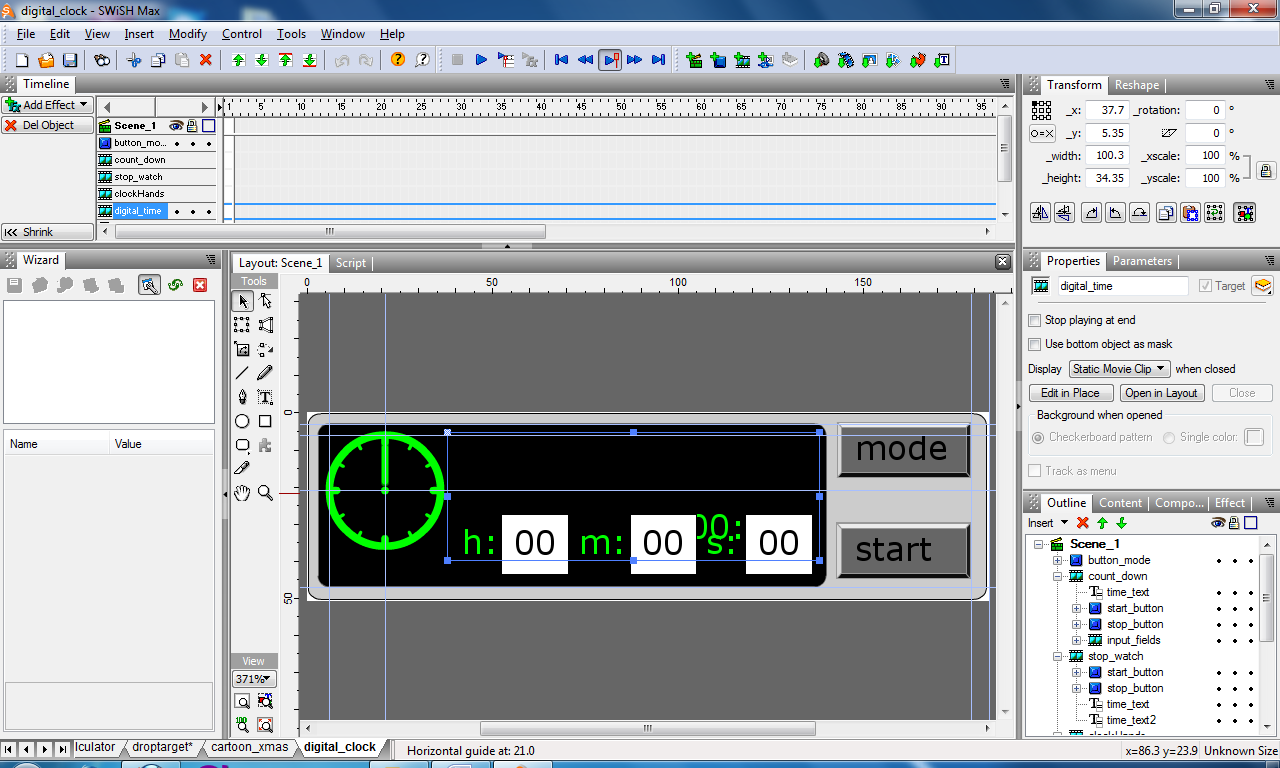
* Ưu điểm
* Công cụ có giao diện và cách sử dụng đơn giản và tiện dụng.
* Hỗ trợ sẵn hệ thống quản lý user và các template page.
* Khuyết điểm
* Dữ liệu dùng để hiển thị là dữ liệu tĩnh. Do đó ứng dụng tạo ra không thể liên kết dữ liệu động.
* Đa số các control và widget đều chỉ hoạt động độc lập.
* Giá của các gói dịch vụ khá cáo.

### SWiSH Max

Website: <http://www.swishzone.com>

* Tổng quan hệ thống phần mềm

SWiSH Max là một sản phẩm của SWiSHzone.com Pty Ltd - công ty chuyên cung cấp công cụ để phát triển Flash animation. Được thành lập vào năm 1999, SWiSH Max là sản phẩm chủ lực của SWiSHzone.com. SWiSH Max là công cụ thiết kế Flash chuyên nghiệp và có tính linh động cao. Hiện SWiSH Max đã được phát triển tới phiên bản 3.0.



Hình ‑ Giao diện của phần mềm SWiSH Max 3

Các tính năng:

SWiSH Max cung cấp công cụ cho phép người sử dụng tạo ra các đối tượng (shape, button…) và drag drop các đối tượng đó trên giao diện thiết kế. Đồng thời cho phép chọn lựa từ những hiệu ứng được cung cấp sẵn (hơn 350 hiệu ứng như phát nổ, quay 3D, sóng…) hoặc tạo ra các hiệu ứng bằng việc kết hợp những hiệu ứng sẵn có để gán cho đối tượng đó.SWiSH Max hỗ trợ việc lập trình với ngôn ngữ script để điều khiển hoạt động của các đối tượng và hiệu ứng.

Ngôn ngữ script sử dụng trong SWiSH Max là SWiSH script, là một dạng biến thể của ActionScript. SWiSH script hỗ trợ hầu hết tất cả các chức năng có trong ActionScript 2.0, và riêng trong SWiSH Max 3 hỗ trợ tất cả các lớp có trong ActionScript 2.0.

Sau khi thiết kế xong, chọn chức năng Export để xuất ra định dạng flash (\*.swf).

* Giá của phần mềm SWiSH Max 3 (cập nhật ngày 05/07/2010):

|  |  |
| --- | --- |
| **Tên sản phẩm** | **Giá** |
| SWiSH miniMax 3 | $69.95 |
| SWiSH Max 3 | $149.95 |

Bảng ‑ Bảng giá của phần mêm SWiSH Max 3

* Ưu khuyết điểm

Sau đây là những ưu khuyết điểm mà nhóm chúng em nhận xét khi sử dụng công cụ này:

* Ưu điểm
* Công cụ có giao diện và cách sử dụng đơn giản và tiện dụng.
* Có số lượng template khá lớn được cung cấp trên trang chủ của SWiSHzone.com.
* Khả năng tùy biến cao.
* Khuyết điểm
* Người sử dụng cần phải có hiểu biết về ActionScript nếu muốn điều khiển hoạt động của các thành phần trong trang web.
* Giá của sản phẩm không rẻ.

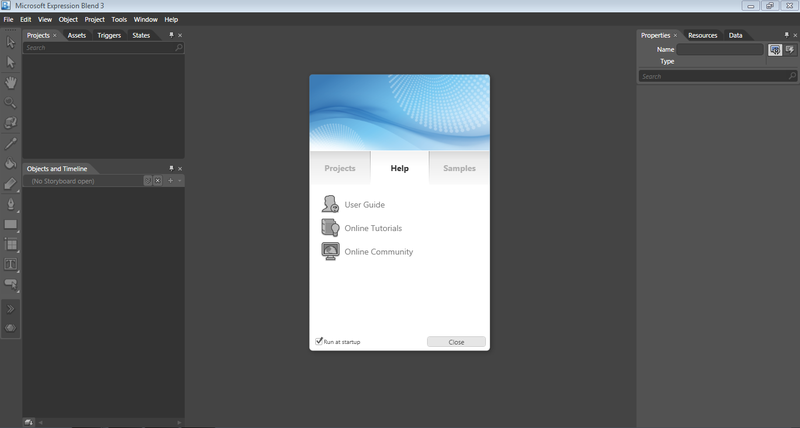
### Microsoft Expression Blend

Website: <http://www.microsoft.com/expression/products/blend_overview.aspx>

* Tổng quan hệ thống phần mềm

Microsoft Expression Blend là một công cụ thiết kế giao diện người dùng được phát triển bởi Microsoft để tạo giao diện đồ họa cho các trang web và các ứng dụng desktop. Đây là công cụ được dùng để thiết kế giao diện XAML cho ứng dụng WPF (Window Presentation Foundation) và ứng dụng Silverlight. Công cụ này là một trong những ứng dụng được cung cấp trong bộ phần mềm ứng dụng Microsoft Expression Studio.

Phiên bản đầu tiên của công cụ này được phát hành miễn phí vào tháng 1 năm 2007. Sau đó được đưa trở thành phần mềm thương mại vào cuối tháng 4 năm 2007. Phiên bản thứ hai của công cụ bắt đầu hỗ trợ phát triển ứng dụng Silverlight, nhưng tại phiên bản này chỉ hỗ trợ Silverlight 1.0, không hỗ trợ Silverlight 2.0. Sau đó, phiên bản thứ ba được phát hành vào tháng 7 năm 2009. Hiện nay, công cụ này được phát triển lên phiên bản tư, vừa được phát hành vào tháng 6 năm 2010. Trong phiên bản mới này, công cụ hỗ trợ Silverlight 4 đồng thời cũng tương thích với các phiên bản Silverlight trước đó.



Hình ‑ Giao diện của Microsoft Expression Blend 3

Các tính năng:

Hỗ trợ thiết kế giao diện một cách trực quan. Ngoài việc hỗ trợ kéo thả control một cách dễ dàng, công cụ còn cung cấp một PropertyGrid rất sinh động cho người dùng điều chỉnh.

Cho phép gán các hiệu ứng cho một control. Đồng thời hỗ trợ thiết kế các hiệu ứng mới ngay trên giao diện.

Có thể mở rộng các hiệu ứng và control đưa từ bên ngoài vào.

Cho phép xử lý sự kiện bằng cách viết mã nguồn.

Công cụ còn hỗ trợ việc liên kết (binding) dữ liệu từ một tập tin xml khá mạnh mẽ.

* Giá của phần mềm Expression Blend 4 (cập nhật ngày 05/07/2010):

|  |  |
| --- | --- |
| **Tên gói phần mềm** | **Giá** |
| Expression Studio 4 Ultimate | $599.95  $349.95 (upgrade từ phiên bản trước) |

Bảng ‑ Bảng giá của phần mêm Microsoft Expression Blend 4

* Ưu khuyết điểm

Sau đây là những ưu khuyết điểm mà nhóm chúng em nhận xét khi sử dụng công cụ này:

* Ưu điểm
* Công cụ có giao diện và cách sử dụng đơn giản và tiện dụng.
* Hỗ trợ việc thiết kế rất trực quan.
* Hỗ trợ tạo ra các hiệu ứng một cách dễ dàng đối với hiệu ứng không quá phức tạp.
* Có khả năng mở rộng số lượng control từ bên ngoài.
* Khuyết điểm
* Người sử dụng cần phải có hiểu biết về lập trình (C#, VB.net…) để có thể xử lý sự kiện và phối hợp hoạt động của các control.
* Giá của công cụ quá cao.

## Bảng so sánh chức năng

### Bảng so sánh chức năng các công cụ Mashup

|  | **iGoogle** | **Dapper** |
| --- | --- | --- |
| Môi trường hoạt động | Web | Web |
| Giá | Miễn phí | Miễn phí |
| Các tính năng | | |
| Cho phép rút trích và chọn lựa dữ liệu từ trang web bất kỳ |  | ✓ |
| Định dạng dữ đầu ra |  | XML, RSS, Google Map… |
| Hỗ trợ tạo Widget | HTML Widget | Flash Widget |
| Hỗ trợ drag drop Widget | ✓ |  |
| Tạo hiệu ứng cho Widget |  |  |
| Phối hợp xử lý giữa các Widget |  |  |

Bảng ‑ Bảng so sánh các chức năng của các công cụ đã khảo sát

### Bảng so sánh chức năng các công cụ RIA Web Designer

|  | **Wix** | **Moonfruit** | **SWiSH Max 3** | **Expression Blend 4** |
| --- | --- | --- | --- | --- |
| Môi trường hoạt động | Web | Web | PC | PC |
| Ứng dụng tạo ra sử dụng công nghệ | Flash | Flash | Flash | Silverlight |
| Đối tượng sử dụng | Người dùng bình thường | Người dùng bình thường | Người có hiểu biết về lập trình | Người có hiểu biết về lập trình |
| Phiên bản hiện tại |  | 4 | 3.0 | 4 |
| Giá | Tùy thuộc vào loại tài khoản Premium | Tùy thời gian và gói dịch vụ | $69.95 - $149.95 | $349.95 - $599.95 |

|  | **Wix** | **Moonfruit** | **SWiSH Max 3** | | **Expression Blend 4** |
| --- | --- | --- | --- | --- | --- |
| Giao diện người dùng | | | | | |
| Giao diện thân thiện, dễ dùng | ✓ | ✓ | ✓ | | ✓ |
| Hỗ trợ drag drop trên giao diện thiết kế | ✓ | ✓ | ✓ | | ✓ |
| Điều chỉnh màu sắc, kích thước trực quan | ✓ | ✓ | ✓ | | ✓ |
| Điều chỉnh các thuộc tính khác |  |  |  | | ✓ |
| Tính năng | | | | | |
| Hỗ trợ các Transition và Effect | ✓ | ✓ | | ✓ | ✓ |
| Tạo ra các hiệu ứng từ giao diện |  |  | |  | ✓ |
| Hỗ trợ liên kết (binding) dữ liệu trực quan |  |  | |  | ✓ |
| Liên kết dữ liệu động |  |  | | Phải viết script | Phải viết mã nguồn |
| Cung cấp sẵn một số Widget | ✓ | ✓ | |  |  |
| Cung cấp các template | ✓ | ✓ | | ✓ |  |
| Phối hợp xử lý giữa các thành phần | Do bản thân công cụ tự thực hiển, người dùng không tùy biến được | Do bản thân công cụ tự thực hiên, người dùng không tùy biến được | | ✓ | ✓ |
| Viết mã script để điều khiển các thành phần |  |  | | Sử dụng SWiSH script | Sử dụng C#, VB.net… |

|  | **Wix** | **Moonfruit** | **SWiSH Max 3** | | **Expression Blend 4** |
| --- | --- | --- | --- | --- | --- |
| Tính năng mở rộng | | | | | |
| Hỗ trợ hosting | ✓ | ✓ | |  |  |
| Xuất kết quả ra file |  |  | | ✓ | ✓ |
| Hỗ trợ sẵn hệ thống quản lý user cho website |  | ✓ | |  |  |

Bảng ‑ Bảng so sánh các chức năng của các công cụ RIA Designer đã khảo sát

## Kết luận

Trong chương này, chúng em đã trình bày hai hệ thống Mashup về giao diện là: ***iGoogle*** và ***Dapper***; bốn công cụ RIA Web Designer là: ***Wix***, ***Moonfruit*** và ***SWiSH Max 3***, ***Microsoft Expression Blend 4***. Thông qua quá trình khảo sát, chúng em đã lập bảng so sánh các chức năng của từng công cụ.

Qua bảng so sánh chức năng cho thấy tất cả các công cụ đều có khả năng hỗ trợ thiết kế trực quan và mỗi công cụ có những ưu khuyết điểm riêng. Hầu hết các công cụ đều để phục vụ việc xây dựng ứng dụng Flash, chỉ riêng ***Microsoft Expression Blend 4*** là hỗ trợ xây dựng ứng dụng Silverlight. Điều đó cho thấy công cụ hỗ trợ *thiết kế nhanh* ứng dụng **Silverlight** vẫn còn *khá ít*.

Hai công cụ ***Wix*** và ***Moonfruit*** hoạt động trên nền web nên có khả năng triển khai rộng rãi, sử dụng ở bất kỳ đâu có Internet. Hai công cụ còn lại là ***SWiSH Max 3*** và ***Microsoft Expression Blend 4*** tuy không thể sử dụng mọi nơi nhưng lại có khả năng hỗ trợ liên kết dữ liệu động cùng với khả năng cho phép điều khiển phối hợp xử lý giữa các thành phần trong ứng dụng web. Tuy nhiên, ngoài Wix có giá thành tương đối rẻ thì các công cụ còn đều giá thành khá cao. Qua đó, chúng em đã xác định được những yêu cầu mà hệ thống công cụ phần mềm cần xây dựng, đó là:

* Một yêu cầu bắt buộc phải có đối với công cụ đó là khả năng hỗ trợ thiết kế một cách trực quan qua các thao tác drag drop
* Cho phép điều chỉnh các thuộc tính một cách dễ dàng
* Hỗ trợ thay đổi và điều chỉnh hiệu ứng cho các control
* Hỗ trợ liên kết dữ liệu một cách trực quan trên giao diện
* Hỗ trợ phối hợp xử lý giữa các thành phần trong ứng dụng mà không cần yêu cầu người dùng biết kiến thức lập trình
* Có khả năng mở rộng số lượng control và hiệu ứng
* Công cụ cần dễ dùng đó với người dùng không có kiến thức lập trình

Ngoài ra, nhóm chúng em còn xác định những yêu cầu sẽ là hướng phát triển của công cụ như:

* Cho phép tạo ra hiệu ứng mới ngay trên giao diện thiết kế từ những hiệu ứng đã có
* Hỗ trợ lập trình cho người dùng có kiến thức về lập trình để điều khiển các thành phần trong ứng dụng hoặc xử lý dữ liệu
* …

Từ đây chúng em có đủ cơ sở phân tích các vấn đề, đề ra các giải pháp tương ứng cũng như xây dựng nên các qui trình cần thiết cho hệ thống công cụ hỗ trợ thiết kế ứng dụng RIA từ các hệ thống website đã có.

# Một số vấn đề và giải pháp hỗ trợ thiết kế trực quan

*✍ Nội dung chương này sẽ trình bày về các vấn đề khi phát triển công cụ hỗ trợ việc thiết kế trực quan ứng dụng web. Từ đó quyết định các giải pháp để xây dựng công cụ.*

## Giới thiệu chung

Trong quá trình xây dựng công cụ hỗ trợ thiết kế trực quan ứng dụng web, nhóm chúng em đã gặp không ít vấn đề cần phải giải quyết. Với mỗi vấn đề, nhóm đã cố gắng đưa ra giải pháp mà chúng em đánh giá là tốt. Sau đây là năm nhóm vấn đề lớn mà nhóm chúng em đã gặp phải và giải quyết:

* Hỗ trợ thiết kế trực quan về vị trí và kích thước (mục 3.2)
* Lấy, gán và hiển thị động danh sách các thuộc tính (mục 3.3)
* Vấn đề control và hiệu ứng (mục 3.4)
* List và nguồn dữ liệu (mục 3.5)
* Phối hợp xử lý giữa các control (mục 3.6)

## Hỗ trợ thiết kế trực quan về vị trí và kích thước

### Hỗ trợ thiết kế vị trí control một cách trực quan

* Vấn đề

Làm thế nào để ta có thể xây dựng được công cụ có khả năng cho người dùng xác định vị trí của control một cách trực quan. Điều đó nghĩa là công cụ cần hỗ trợ cho người dùng xác định vị trí của control chỉ bằng các thao tác drag drop trên giao diện thiết kế. Vấn đề đặt ra là cần xác định người dùng sẽ thực hiện những thao tác nào, và mục đích của thao tác đó là gì?

* Giải pháp

Qua việc phân tích quá trình drag drop một control trên giao diện thiết kế của người dùng, nhóm nhận thấy ra quá trình này diễn ra qua ba bước:

* Bước 1: nhấn chuột trái để bắt đầu quá trình drag drop
* Bước 2: giữ chuột trái và rê chuột đến vị trí cần đặt control
* Bước 3: thả chuột trái để kết thúc quá trình drag drop

Với quá trình vừa được xác định, nhóm đã đề ra được giải pháp tổng quát có thể cài đặt ở bất kỳ công nghệ nào mà không phụ thuộc vào công nghệ Silverlight. Giải pháp đó như sau:

* Bắt sự kiện nhấn chuột trái (*MouseLeftButtonDown*) của control: khi sự kiện này được phát sinh, sẽ bật cờ *isCaptured* để đánh dấu là đang giữ chuột trái đồng thời lấy vị trí tương đối của con trỏ chuột đối với control (*clickPoint*).
* Bắt sự kiện di chuyển chuột (*MouseMove*) của control: khi sự kiện được phát sinh, kiểm tra cờ *isCaptured* có được bật không? Nếu có bật, lấy vị trí tương đối con trỏ chuột đối với vùng thiết kế (*currPoint*), sau đó tính lại vị trí của control và cập nhật trên vùng thiết kế. Vị trí mới của control được tính theo công thức sau:

x = currPoint.X – clickPoint.X

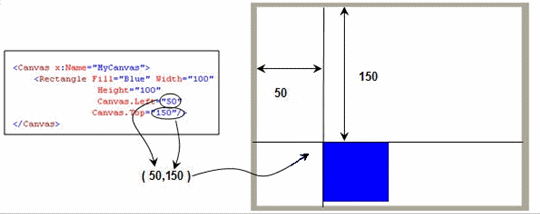
y = currPoint.Y – clickPoint.Y

* Bắt sự kiện thả chuột trái (*MouseLeftButtonUp*) của control: khi sự kiện này được phát sinh, sẽ tắt cờ *isCaptured* để đánh dấu là không còn giữ chuột trái.
* **Cài đặt trên Silverlight**

Việc cài đặt trên Silverlight hoàn toàn tương tự với giải pháp được nêu trên. Tuy nhiên, Silverlight hỗ trợ sẵn ba loại Layout Manager (*Canvas, Grid* và *StackPanel)* và mỗi loại có đặc tính riêng. Do đó, cần chọn loại Layout Manager nào để làm vùng thiết kế. Yêu cầu là loại Layout Manager được chọn cần phải hỗ trợ xác định vị trí control một cách đơn giản.

* Canvas:

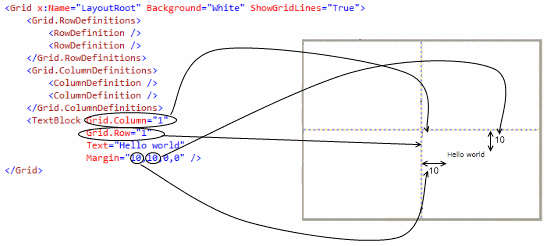
Là loại layout manager hỗ trợ việc đặt vị trí của control bằng cách dùng tọa độ cụ thể. Vị trí của một control trong Canvas được xác định thông qua hai thuộc tính kèm theo (***Attached Property*)** là *Canvas.Left*, *Canvas.Top*. Do đó, việc xác định vị trí của control rất đơn giản và trực quan. Ngoài ra Canvas còn hỗ trợ thêm một Attached Property là *Canvas.ZIndex*, đây là thuộc để xác định độ sâu của layer mà control đang được đặt (mặc đinh control nằm ở layer 0).



Hình ‑ Cách xác định vị trí của control trên Canvas

* Grid:

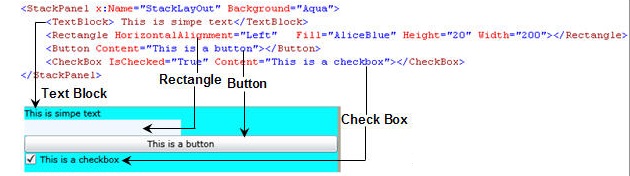
Là loại layout manager cho phép sắp xếp các control theo dạng bảng gồm nhiều dòng, nhiều cột. Vị trí dòng hoặc cột của control trên Grid được chỉ định thông qua hai **Attached Property** là *Grid.Row*, *Grid.Column*. Vị trí control được xác định dựa vào thuộc tính *Margin* của control và thuộc tính *Grid.Row,* *Grid.Column*.



Hình ‑ Cách xác định vị trí của control trên Grid

* StackPanel:

Là loại layout manager cho phép sắp xếp các control theo chiều ngay hay chiều dọc. Vị trí của control phụ thuộc vào kích thước của các control khác trong StackPanel.



Hình ‑ Cách xác định vị trí của control trên StackPanel

Qua việc tìm hiểu về các Layout Manager, nhóm nhận thấy Canvas là loại layout có thể đáp ứng được việc xác định vị trí một cách dễ dàng và còn hỗ trợ khả năng sắp xếp control theo độ sâu của layer.

### Xác định control được chọn trên giao diện thiết kế

* Vấn đề

Làm sao để có thể biết được khi nào một control được chọn. Giải pháp đơn giản được chúng em đưa ra là dựa vào sự kiện click chuột trái lên control đó để xác định. Nghĩa là khi một control được click vào đồng nghĩa với việc control đó sẽ được chọn. Nhưng nếu trong control có chứa các control có thể tương tác bằng thao tác click chuột (như button, checkbox, combobox…), lúc đó khi người dùng click vào vị trí của những control có thể tương tác bằng chuột đó thì sự kiện click chuột trái sẽ không phát sinh ở control chứa chúng.

Vấn đề được đặt ra là làm sao để có thể xác định được control được chọn khi người dùng click vào bất kỳ vị trí nào trên control đó?

* Giải pháp

Qua quá trình tìm hiểu, nhóm đã đề ra được 3 giải pháp:

* Giải pháp 1

Chặn tất cả các sự kiện liên quan đến click chuột được gửi đến các control có thể tương tác được bằng click chuột được chứa trong control đang xét. Sau đó, phát sinh ra sự kiện focus cho control chứa những control đó.

* Ưu điểm
* Không có
* Khuyết điểm
* Việc xử lý khá phức tạp.
* Gây chậm hệ thống do phải tiến hành chặn sự kiện.
* Không khả thi do trên thực tế lúc runtime, chương trình không thể nào biết được trong control đang xét có chứa bao nhiêu control tương tác được bằng click chuột.
* Giải pháp 2

Không hiển thị trực tiếp control lên giao diện, tiến hành capture lại control này và dùng hình ảnh đó để hiện thị lên giao diện design. Khi có sự thay đổi về vị trí, tọa độ, màu sắc… của control thì sẽ tiến hành capture lại và cập nhật hình mới. Với giải pháp này, thực chất người dùng chỉ tương tác với ảnh của control chứ không tương tác với control thật.

* Ưu điểm
* Việc xử lý không quá phức tạp, thể hiện trực quan hình ảnh của control tại mỗi thời điểm.
* Khuyết điểm
* Do gần như phải liên tục capture lại control nên sẽ làm chậm việc xử lý của hệ thống.
* Khi càng nhiều control trên màn hình cần phải dùng càng nhiều hình nên bộ nhớ sử dụng sẽ rất lớn.
* Người dùng không thể tương tác được với các control bên trong qua thao tác click chuột lúc thiết kế.
* Giải pháp 3

Sử dụng một hình chữ nhật trong suốt để phủ lên trên control. Hình chữ này được chúng em gọi là *ProxyControl*. Vị trí và kích thước của *ProxyControl* này sẽ phục thuộc vào vị trí và kích thước của control và ngược lại. Nghĩa là khi có sự thay đổi về vị trí hoặc kích thước của một trong hai thì sẽ cập nhật đồng thời cả hai. Với giải pháp này, khi người ta click chuột để chọn control, thực chất là đang tương tác với *ProxyControl* chứ không tương tác với control.

* Ưu điểm
* Việc xử lý đơn giản.
* Có thể thêm vào đường viền xung quanh để thể hiện control đó đang được chọn bằng cách vẽ thêm cho ProxyControl đường viền này.
* Khuyết điểm
* Người dùng không thể tương tác được với các control bên trong qua thao tác click chuột lúc thiết kế.
* Tốn bộ nhớ do phải dùng thêm ProxyControl.

🗒 Nhận xét

Với ba giải pháp được đề xuất thì chúng em nhận thấy chỉ có giải pháp 2 và giải pháp 3 là có khả thi. Qua việc phân tích hai giải pháp này, nhóm có nhận xét như sau:

* Nếu thực hiện theo giải pháp 2 thì việc xử lý của hệ thống sẽ bị chậm do phải trả thêm chi phí cho việc capture control và cần phải dùng nhiều bộ nhớ hơn để chứa thêm hình ảnh của control.
* Còn với giải pháp 3 tuy phải dùng thêm bộ nhớ để lưu trữ ProxyControl nhưng chi phí cho việc này thì nhỏ hơn so với giải pháp 2, đồng thời giải pháp này còn hỗ trợ thêm việc hiển thị trực quan những control đang được chọn thông qua đường viền bên ngoài.

Vì thế nhóm đã quyết định chọn sử dụng giải pháp 3 bởi những ưu điểm của giải pháp này. Đồng thời, giải pháp này cũng không phụ thuộc vào công nghệ Silverlight, nghĩa là đây là giải pháp tổng quát có thể sử dụng bất kỳ công nghệ RIA nào cũng có thể cài đặt.

### Hỗ trợ thiết kế kích thước control một cách trực quan

* Vấn đề

Với một control đã được chọn, làm thế nào để có thể cho người dùng xác định, điều chỉnh kích thước của control đó một cách trực quan. Nếu chỉ cho người dùng điều chỉnh kích thước trực tiếp thông qua việc chỉnh giá trị của thuộc tính *Width* và *Height* của control thì sẽ rất bất tiện và không trực quan. Vấn đề đặt ra là chương trình phải thể hiện việc xác định kích thước control một cách trực quan trên giao diện thiết kế chỉ bằng các thao tác chuột đơn giản?

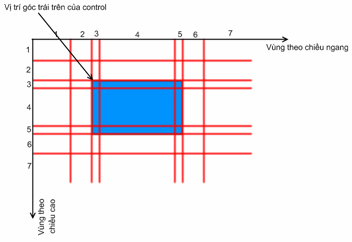
* Giải pháp

Qua việc tìm hiểu “việc xác định kích thước control trên giao diện bằng các thao tác click chuột đơn giản”, nhóm chúng em nhận thấy quá trình resize này diễn ra qua bốn bước và với mỗi bước thì nhóm cũng đưa ra giải pháp tổng quát để có thể hiện thực được bước đó. Sau đây là các bước và giải pháp tổng quát:

* ***Bước 1***: Xác định nơi nào có thể resize được và chiều resize. Đây là bước thể hiện cho người dùng khi nào có thể thay đổi được kích thước của control đang được chọn và việc thay đổi sẽ tác động đến chiều rộng (*Width*) hoặc chiều cao (*Height*) của control hoặc đồng thời cả hai thông qua việc bắt sự kiện di chuyển chuột (*MouseMove*). Việc đó nghĩa là cần thể hiện lại hình dạng của con trỏ chuột dựa vào vị trí của chuột với vị trí và kích thước của control đang được chọn.

Vấn đề ở đây là nếu chỉ bắt sự kiện MouseMove trên control được chọn thì khi di chuyển con trỏ chuột ra khỏi phạm vi của control thì sẽ không thể bắt được sự kiện này nên còn bắt sự kiện MouseMove của vùng thiết kế chứa control.

Nhóm đã chia những vùng xung quanh và cả vùng chứa control ra làm *bảy vùng theo chiều ngang* và *bảy vùng theo chiều dọc* được thể hiện ở Hình 3‑4.



Hình ‑ Phân vùng control để xác định khả năng resize

Những vùng (vị trí vùng theo chiều ngang, vị trí vùng theo chiều dọc) có thể resize và chiều tác động như sau:

* Resize theo chiều cao: vùng (4,2), (4,3), (4,5), (4,6).
* Resize theo chiều ngang: vùng (2,4), (3,4), (5,4), (6,4).
* Resize theo cả 2 chiều: vùng (2,2), (2,3), (3,2), (3,3), (2,5), (2,6), (3,5), (3,6), (5,2), (5,3), (6,2), (6,3), (5,5), (5,6), (6,5), (6,6).
* ***Bước 2***: Tại những vùng có thể resize được, khi người dùng nhấn chuột, có nghĩa là người dùng đã chọn bắt đầu resize và chiều tác động đến kích thước control. Ở bước này, ta sẽ bật cờ tên *isResizing* lên để đánh dấu đang resize.
* ***Bước 3:*** Người dùng giữ chuột trái và rê chuột để thay đổi kích thước. Tại bước này, sẽ dựa vào vùng mà người dùng bắt đầu resize để thay đổi kích thước và vị trí (nếu cần) của control.
* ***Bước 4:*** Người dùng thả chuột trái ra có nghĩa là đã hoàn tất việc resize. Tại bước này, tắt cờ *isResizing* để đánh dấu không còn ở trạng thái resize.

### Dock các control

* Vấn đề

Khi người dùng sử dụng chương trình để tạo ứng dụng thì sẽ có nhu cầu cho một control chiếm cố định một vùng nào đó, như Menu sẽ chiếm vùng trên đầu… Đó chính là vấn đề Dock.

* Giải pháp

Qua tìm hiểu thì nhóm em biết được Silverlight Toolkit - thư viện control chính thức cho Silverlight - có hỗ trợ control *DockPanel*. Tuy nhiên có một điều rất đáng tiếc là DockPanel hỗ trợ bốn kiểu Dock (Left/Top/Right/Bottom) nhưng lại không hỗ trợ kiểu Dock None (không Dock) cho những control có thể di chuyển tự do. Do đó, nhóm em quyết định cài đặt một lớp *DockCanvas*, bằng cách mở rộng thêm khả năng Dock từ lớp Canvas của Silverlight.

* **Cách thực hiện:**

Việc cài đặt DockCanvas khá đơn giản, chỉ cần bổ sung hàm cập nhật để cập nhật control khi kích thước DockCanvas thay đổi hoặc ZIndex, Dock của control nằm trong DockCanvas thay đổi.

Hàm cập nhật control được cài đặt như sau:

* Bước 1: sắp xếp các control trong Canvas theo thứ tự tăng dần của ZIndex.
* Bước 2: dùng một biến *remainRect* kiểu *Rect* để lưu lại vùng hình chữ nhật còn trống. Lúc đầu thì *remainRect* sẽ bằng với Canvas.
* Bước 3: Lần lượt duyệt qua các control và xét giá trị của thuộc tính Dock:
  + Nếu là None thì không cần làm gì cả.
  + Nếu là Fill thì chiều rộng và chiều cao bằng chiều rộng và chiều cao của Canvas. Vị trí Left, Top là (*remainRect.Left*, *remainRect.Top*).
  + Nếu là Left: chiều rộng của *remainRect* giảm đi, *remainRect.Left* tăng lên, chiều cao của control bằng chiều cao của *remainRect*. Vị trí Left, Top của control bằng Left, Top của remainRect.
  + Nếu là Right: chiều rộng của *remainRect* giảm đi, chiều cao của control bằng chiều cao của *remainRect*. Vị trí Left, Top của control bằng Left, Top của remainRect.
  + Nếu là Top, chiều cao của *remainRect* giảm đi, *remainRect.Top* tăng lên, chiều rộng của control bằng chiều rộng của *remainRect.* Vị trí Left, Top của control bằng Left, Top của remainRect.

## Lấy, gán và hiển thị động danh sách các thuộc tính

### Lấy và gán giá trị các thuộc tính của control lúc runtime

* Vấn đề

Khi người dùng sử dụng chương trình để tạo ứng dụng thì sẽ có nhu cầu thay đổi các thuộc tính của control (màu sắc, font chữ…). Nếu các control này là những control có sẵn, ta biết được bên trong control có những thuộc tính nào thì việc xử lý sẽ đơn giản. Do chương trình cho phép bổ sung các control dưới dạng dll nên cần phải lấy và gán giá trị cho thuộc tính của control lúc runtime.

* Giải pháp

Qua tìm hiểu thì nhóm em biết được ba giải pháp cho vần đề này:

* Giải pháp 1

Định nghĩa trước một lớp trừu tượng, trong đó chứa các thuộc tính cần dùng, những control mới thêm vào sẽ kế thừa từ lớp trừu tượng này. Khi đó, ta sẽ lấy và gán giá trị thông qua lớp trừu tượng.

* Ưu điểm
* Cài đặt đơn giản.
* Khuyết điểm
* Control mới chỉ có những thuộc tính của lớp trừu tượng, không thể định nghĩa thêm thuộc tính mới.
* Control mới bắt buộc phải kế thừa từ lớp trừu tượng.
* Giải pháp 2

Sử dụng kỹ thuật Reflection. Với cách này ta có thể liệt kê tất cả các thuộc tính có trong control, lấy và gán giá trị cho những thuộc tính đó lúc runtime.

* Ưu điểm
* Tính linh động rất cao, control thêm vào có thể có thuộc tính bất kỳ, không bị giới hạn.
* Control mới không bị ràng buộc phải kế thừa từ lớp cơ sở.
* Khuyết điểm
* Không được chọn lựa thuộc tính cho phép tùy biến.
* Mặc dù control thêm vào có thể có nhiều thuộc tính, nhưng kiểu của các thuộc tính phải nằm trong những kiểu được chương trình hỗ trợ vì nếu không sẽ không có giao diện nhập giá trị cho thuộc tính.
* Giải pháp 3

Vẫn sử dụng kỹ thuật Reflection ở giải pháp 2 nhưng kết hợp thêm ý tưởng giải pháp 1. Cụ thể như sau: ta có một lớp cơ sở, các control sẽ kế thừa lại. Trong lớp cơ sở có một biến *parameterNameList*, chứa danh sách tên những thuộc tính của control sẽ cho người dùng thay đổi giá trị. Ta không lấy tất cả thuộc tính của control như giải pháp 2, mà chỉ lấy những thuộc tính có tên trong *parameterNameList*.

* Ưu điểm
* Tính linh động cao, control thêm vào có thể có thuộc tính bất kỳ, không bị giới hạn.
* Người viết control có thể quyết định là thuộc tính nào sẽ cho người dùng hiệu chỉnh.
* Khuyết điểm
* Phải kế thừa từ lớp cơ sở.
* Kiểu của các thuộc tính phải nằm trong những kiểu được chương trình hỗ trợ vì nếu không sẽ có có giao diện nhập giá trị cho thuộc tính.

🗒 Nhận xét :

Giải pháp 1 làm cho khả năng tùy biến các control rất thấp vì chỉ được dùng các thuộc tính qui định sẵn. Giải pháp 2 có ưu điểm rất lớn là không cần kế thừa từ lớp cơ sở nhưng vì liệt kê tất cả các thuộc tính, không cho chọn lựa nên danh sách thuộc tính rất dài, nhiều cái không dùng đến. Do đó, nhóm em quyết định chọn giải pháp 3 là giải pháp dung hòa những ưu và khuyết của giải pháp 1 và 2.

* **Cách thực hiện:**

*Cách lấy và gán giá trị cho một thuộc tính (property) của đối tượng bằng kỹ thuật reflection:*

* Bước 1: lấy *Type* của đối tượng bằng hàm *GetType*().
* Bước 2: lấy đối tượng *PropertyInfo* của property cần lấy hoặc gán giá trị bằng cách gọi hàm *GetProperty* với tham số là tên của property cần lấy thông tin.
* Bước 3: trong đối tượng PropertyInfo có cung cấp hàm *GetValue* để lấy giá trị của property và hàm *SetValue* để gán giá trị cho property.
  + Hàm GetValue nhận vào hai tham số:
    - Tham số 1: đối tượng chứa property cần lấy giá trị
    - Tham số 2: nếu giá trị của property thuộc kiểu dữ liệu list hoặc collection thì sẽ là mảng chứa chỉ mục của phần tử cần lấy giá trị. Ngược lại, truyền null cho tham số này.
    - *Ví dụ 1*: để lấy giá trị của thuộc tính Width của một đối tượng btn thuộc kiểu dữ liệu Button, ta sẽ gọi hàm như sau:

double width = (double) btn.GetType().GetProperty("Width").GetValue(btn, null);

* + - *Ví dụ 2*: để lấy giá trị của phần tử thứ hai trong một đối tượng list thuộc kiểu dữ liệu List, ta sẽ gọi hàm như sau:

int[] index = new int[]{ 2 };

object value = list.GetType().GetProperty("Item").GetValue(list, index);

* + Hàm SetValue nhận vào ba tham số:
    - Tham số 1: đối tượng chứa property cần gán giá trị
    - Tham số 2: giá trị cần gán vào property.
    - Tham số 3: tương tự tham số hai của hàm GetValue
    - *Ví dụ 1*: để gán giá trị của thuộc tính Width của một đối tượng btn thuộc kiểu dữ liệu Button, ta sẽ gọi hàm như sau:

btn.GetType().GetProperty("Width").SetValue(btn, 800, null);

* + - *Ví dụ 2*: để gán giá trị của phần tử thứ hai trong một đối tượng list thuộc kiểu dữ liệu List, ta sẽ gọi hàm như sau:

object value = 5;

int[] index = new int[]{ 2 };

list.GetType().GetProperty("Item").GetValue(list, value, index);

Cách làm cơ bản là vậy. Nhưng do mỗi kiểu dữ liệu cần có giao diện nhập liệu riêng, và có rất nhiều kiểu dữ liệu nên việc xử lý giao diện khá phức tạp. Do đó, nhóm chúng em không làm từ đầu mà phát triển thêm dựa trên project PropertyGrid opensource ]. Mặc dù PropertyGrid chỉ hỗ trợ một vài kiểu dữ liệu cơ bản nhưng được viết theo một cấu trúc rất linh hoạt, có thể bổ sung thêm các kiểu dữ liệu mới dễ dàng. PropertyGrid sẽ được mô tả chi tiết ở mục 3.3.2.

### Hiển thị động danh sách các thuộc tính ở dạng PropertyGrid

PropertyGrid là control gồm hai cột và nhiều dòng. Mỗi dòng thể hiện một thuộc tính của control. Cột thứ nhất là tên thuộc tính. Cột thứ hai là editor để hiển thị và thay đổi giá trị của thuộc tính, với editor mặc định là *StringEditor*.

*Cách thêm kiểu dữ liệu mới cho PropertyGrid:*

* Cách 1: dành cho kiểu dữ liệu có thể nhập liệu dạng string (như Uri, Thickness...), ta chỉ cần viết thêm một lớp converter kế thừa từ TypeConverter để convert dữ liệu dạng string về kiểu dữ liệu mong muốn, và thêm dòng sau đây vào hàm GetCoreConverterFromCustomType của lớp TypeConverterHelper:

if (typeof(kieudulieu).IsAssignableFrom(type))

return new TenConverter();

Ý nghĩa của dòng trên là nếu gặp thuộc tính có kiểu là kieudulieu thì dùng TenConverter để chuyển từ string về đúng kieudulieu.

* Cách 2: Khi muốn giao diện nhập liệu khác dạng string, cần thực hiện những cài đặt sau: viết thêm một lớp Editor mới, kế thừa từ ValueEditorBase hoặc lớp Editor có sẵn thích hợp. Sau đó, thêm vào hàm GetEditor có ba tham số của lớp EditorService dòng sau đây:

if (typeof(kieudulieu).IsAssignableFrom(propertyType))

return new tenEditor(label, property);

Ý nghĩa của dòng trên là nếu gặp thuộc tính có kiểu là kieudulieu thì sẽ dùng Editor tương ứng là tenEditor.

*Bổ sung Attached Property cho PropertyGrid*:

* Trong Silverlight có một loại thuộc tính rất đặc biệt, đó là ***Attached Property***. Loại thuộc tính được khai báo trong một lớp, nhưng lại đem đi gán (attach) cho đối tượng thuộc lớp khác. Dữ liệu thực sự của thuộc tính được lưu ở đối tượng bị gán thuộc tính, nhưng đối tượng đó lại không thể tự lấy được giá trị của thuộc tính! Muốn lấy được giá trị ta phải biết tên lớp khai báo thuộc tính. Ví dụ về ***Attached Property***, đó là thuộc tính Left của lớp Canvas. Thuộc tính này được khai báo trong lớp Canvas, nhưng được gán cho các control khác (như textbox, button... ). Giá trị Left sẽ được lưu tại control bị gán (textbox, button…) nhưng các control này không truy xuất được giá trị Left.
* Cách lấy giá trị của thuộc tính Attached Property:

obj.GetValue(ownerType.PropertyName, value);

ownerType.GetPropertyName(obj, value);

* Cách gán giá trị cho thuộc tính Attached Property:

obj.SetValue(ownerType.PropertyName, value);

ownerType.SetPropertyName(obj, value);

Với obj là control cần gán giá trị (button, textbox,…), onwerType là lớp khai báo (Canvas), PropertyName là tên thuộc tính (Left)

* PropertyGrid không hỗ trợ sẵn Attached Property vì PropertyGrid chỉ nhận vào đối tượng cần liệt kê thuộc tính, không thể lấy được Attached Property. Nhóm em giải quyết bằng cách bổ sung thêm tham số là cha của đối tượng. Sau đó, lấy hàm get và hàm set bằng cách sau:

MethodInfo get = parentType.GetMethod("Get" + name, BindingFlags.Public | BindingFlags.Static | BindingFlags.FlattenHierarchy);

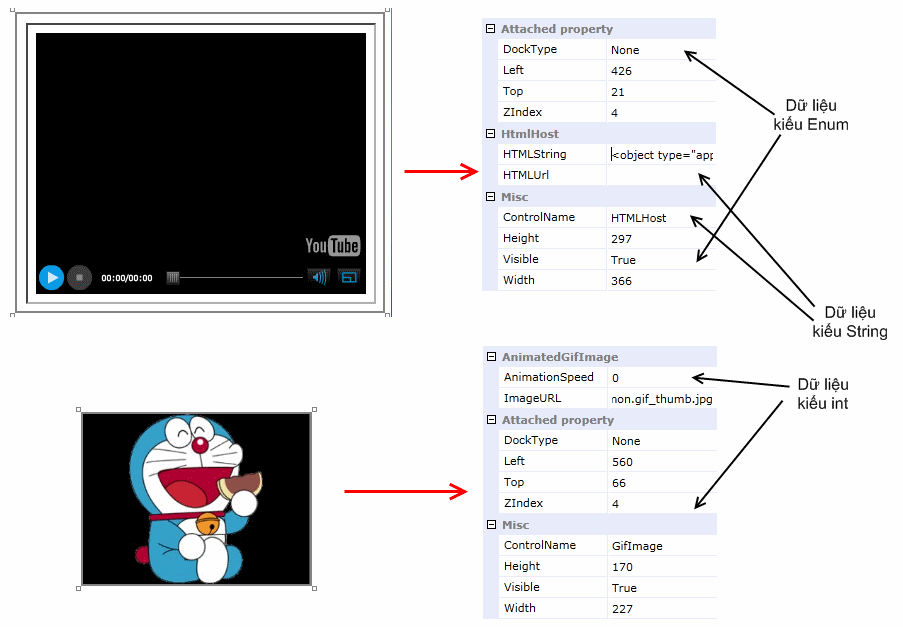
MethodInfo set = parentType.GetMethod("Set" + name, BindingFlags.Public | BindingFlags.Static | BindingFlags.FlattenHierarchy);

Sau khi có hàm get và hàm set thì có thể lấy và gán giá trị cho Attached Property:

value = get.Invoke(null, new object[]{obj })

set.Invoke(null, new object[]{obj, value})

Hình 3‑5 thể hiện giao diện của PropertyGrid với hai control khác nhau. Hình 3‑5 cho thấy khả năng thay đổi động danh sách các thuộc tính được hiển thị trên PropertyGrid tương ứng với control được. Ngoài ra, Hình 3‑5 còn thể hiện khả năng hiển thị và cho phép chỉnh sửa nhiều kiểu dữ liệu khác nhau ở cột editor (kiểu string, kiểu enum, kiểu int…)



Hình ‑ Giao diện PropertyGrid với các control khác nhau

## Vấn đề control và hiệu ứng

### Tách riêng control và hiệu ứng

* Vấn đề

Làm sao để có thể một control sẽ mang trên nó những hiệu ứng và làm sao để một hiệu ứng có thể thể hiện trên một control. Vấn đề đặt ra là có nên tách riêng hiệu ứng ra khỏi control hay không?

* Giải pháp

Với vấn đề “có nên tách riêng hiệu ứng với control hay không”, nhóm chúng em đã đề ra hai giải pháp:

* Giải pháp 1

Hiệu ứng và control sẽ không tách rời nhau, một control sẽ chỉ có những hiệu ứng nhất định trên đó và không thay đổi được. Những hiệu ứng trên control là cố định, có bao nhiêu hiệu ứng và hiệu ứng đó được thực hiện khi nào sẽ do người xây dựng control quyết định.

* Ưu điểm
* Do biết trước control sẽ mang trên nó những hiệu ứng gì nên sẽ dễ kiểm soát được hiệu ứng có hoạt động tốt trên control đó hay không, đồng thời cũng sẽ có thể làm những hiệu ứng đặc thù riêng biệt cho control đó.
* Khuyết điểm
* Không thể thay đổi được hiệu ứng cho một control.
* Nếu muốn dùng một hiệu ứng cho nhiều control khác nhau thì phải cài đặt lại hiệu ứng đó nhiều lần.
* Giải pháp 2

Hiệu ứng và control sẽ được tách riêng như sơ đồ lớp ở Hình 3‑6



Hình ‑ Sơ đồ kế thừa của control và hiệu ứng

Một control mới cần cài đặt sẽ được kế thừa trực tiếp từ *BasicControl* hoặc *BasicListControl*. Sự khác nhau của việc kế thừa từ hai class này là BasicListControl là control dạng danh sách, còn BasicControl có thể là control dạng danh sách hoặc một control thường. Một control sẽ phải override lại hàm:

* *ChangeEffect*: để thay đổi hiệu ứng. Hàm này sẽ được gọi khi cần thay đổi 1 hiệu ứng. Khi đó, bản thân control sẽ biết là cần thay đổi hiệu ứng áp đặt lên control nào chứa bên trong control đó dựa vào tên được truyền vào và tiến hành thay đổi.

Một effect mới cần cài đặt sẽ được kế thừa từ *BasicEffect* hoặc *BasicListEffect*. Sự khác nhau của hai class này cũng tương tự như sự khác nhau của BasicControl và BasicListControl. Đó là BasicListEffect là hiệu ứng cho BasicListControl, còn BasicEffect là hiệu ứng cho BasicControl. Một effect sẽ phải override lại ba hàm gồm:

* *Start* để gọi thực thi hiệu ứng
* *Stop* để ngưng thực thi hiệu ứng.
* *DetachEffect* được gọi lúc gỡ bỏ hiệu ứng, tác dụng của hàm này là để trả lại hiện trạng của control trước khi thực hiện gán hiệu ứng vào.

Tất cả những control hoặc hiệu ứng mới được cài đặt sẽ cần phải override lại cái phương thức kế thừa từ *IBasic*. Những phương thức này là những phương thức để lấy danh sách các thuộc tính cho phép thay đổi giá trị, lấy giá trị và kiểu dữ liệu của những thuộc tính đó, gán lại giá trị cho những thuộc tính đó.

Khi ấy, control sẽ đưa ra những property hiệu ứng (con trỏ hiệu ứng) để cho phép thay đổi hiệu ứng. Mặc định, control đã đưa ra một con trỏ hiệu ứng tên là *MainEffect* dạng BasicEffect, đây là hiệu ứng áp đặt lên control. Trường hợp muốn cho phép điều chỉnh hiệu ứng cho control con bên trong một control, control con phải được kế thừa từ BasicControl hoặc BasicListControl, đồng thời control chứa control con này phải đưa ra một con trỏ hiệu ứng tương ứng.

Việc gán hiệu ứng cho control sẽ được nêu ở các mục:

* 3.4.2 Hiệu ứng BasicEffect
* 3.4.3 Thay đổi hiệu ứng cho control dạng BasicListControl
* 3.4.4 Gán hiệu ứng cho control lúc runtime
* Ưu điểm
* Khả năng tái sử dụng lại hiệu ứng cao. Khi thêm một control vào chương trình, control đó đã có thể sử dụng được các hiệu ứng có sẵn trong chương trình.
* Khả năng mở rộng hiệu ứng cho control cao. Khi thêm một hiệu ứng mới vào chương trình, tất cả những control có thể sử dụng hiệu ứng này đều đã sẵn sàng sử dụng mà không cần phải chỉnh sửa trong control đó.
* Khả năng tùy biến cao cho người sử dụng. Với cùng một loại control có thể tạo ra những control có những hiệu ứng khác nhau.
* Khuyết điểm
* Do không biết trước được các hiệu ứng nên khó kiểm soát được hiệu ứng đó có hoạt động tốt trên control hay không.

🗒 Nhận xét

Qua việc phân tích hai giải pháp đã nêu. Nhóm nhận thấy rằng giải pháp 2 có tính linh động và tính tái sử dụng cao nên nhóm đã quyết định chọn giải pháp này. Ở giải pháp này sẽ tạo được tính chuyên biệt, người xây dựng control sẽ chỉ quan tâm đến việc cài đặt control như thế nào, người xây dựng hiệu ứng sẽ chỉ quan tâm đến việc cài đặt hiệu ứng đó như thế nào.

### Hiệu ứng BasicEffect

* Vấn đề

Qua phân tích, nhóm nhận thấy rằng một hiệu ứng có thể chỉ cần tác động lên chính control cần gán hiệu ứng (như thay đổi màu sắc, độ trong suốt của control…) đã có thể thực hiện hiệu ứng. Nhưng cũng có những hiệu ứng cần thêm một số vật liệu khác vào mới có thể thực hiện hiệu ứng (như cần thêm một hình chữ nhật để phủ lên trên control…). Vấn đề đặt ra là làm sao để hiệu ứng có thể khởi tạo những tham số cần thiết để thực hiện hiệu ứng đó?

* Giải pháp

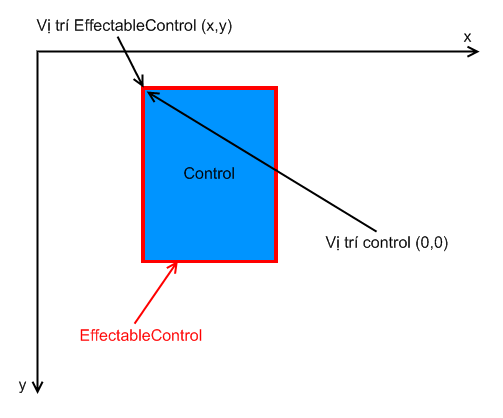
Qua việc phân tích nhóm đã đề ra hai giải pháp để giải quyết vấn đề:

* Giải pháp 1

Hiệu ứng sẽ điều chỉnh các thông số trên control và thêm vào những vật liệu cần thiết để thực hiện hiệu ứng trực tiếp lên Canvas chứa control đó. Vị trí và kích thước cũng những vật liệu cần thêm vào sẽ phải theo vị trí và kích thước của control trên Canvas chứa control đó.

* Ưu điểm
* Không có
* Khuyết điểm
* Nếu control cần gán hiệu ứng nằm dưới một control khác, khi hiệu ứng thêm những vật liệu cần thiết thì phải gán độ sâu layer (ZIndex) của những vật liệu đã thêm vào sao cho không được đè lên control kia.
* Việc xác định vị trí của những vật liệu cần thêm tương đối phức tạp do phải xác định theo vị trí của control trên Canvas đồng thời phải theo vị trí tương đối của vật liệu đó trên control.
* Khi control được xóa khỏi Canvas, đồng thời cần phải xóa tất cả những vật liệu đã thêm ra khỏi Canvas. Nếu xóa sót vật liệu nào đó, vật liệu đó sẽ nằm mãi trên Canvas.
* Giải pháp 2

Không thêm trực tiếp control vào Canvas mà tiến hành đặt control vào một control gọi là *EffectableControl* và đặt control đó vào Canvas. Thực chất EffectableControl chỉ là một Canvas dùng để chứa control, có kích thước bằng với kích thước của control và có màu nên là trong suốt. Hiệu ứng vẫn sẽ điều chỉnh các thông số trên control, nhưng những vật liệu cần thiết để thực hiện hiệu ứng sẽ được thêm vào Canvas của EffectableControl. Do chỉ thêm EffectableControl vào trong Canvas của giao diện thiết kế nên việc xác định Zindex của control bây giờ không còn được xác định thông qua chính control đó nữa, việc xác định này được xác định thông qua Zindex của EffectableControl. Tương tự với vị trí Left, Top của control cũng sẽ được xác định thông qua Left, Top của EffectableControl.



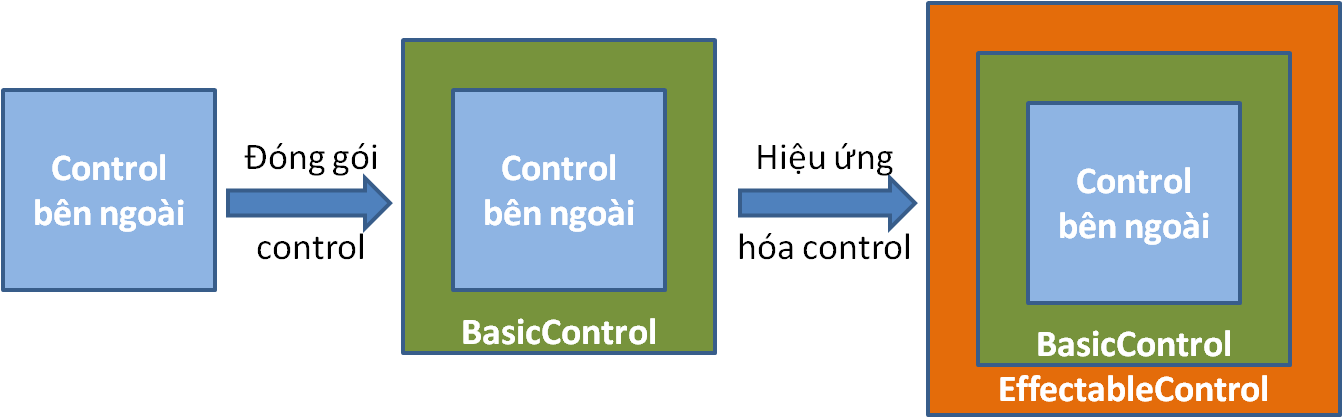
Hình ‑ EffectableControl

* Ưu điểm
* Vấn đề về độ sâu layer (ZIndex) đã được giải quyết một cách đơn giản do tất cả vật liệu và control đều nằm chung trong EffectableControl.
* Việc xác định vị trí của vật liệu chỉ cần xác định bằng vị trí tương đối với control.
* Khi muốn xóa một control, chỉ cần thực hiện một thao tác xóa EffectableControl đó ra khỏi Canvas. Không cần xóa vật liệu vì vật liệu đã nằm trong EffectableControl nên cũng sẽ được xóa theo.
* Khuyết điểm
* Không có

🗒 Nhận xét

Qua việc phân tích hai giải pháp đã nêu. Nhóm nhận thấy rằng giải pháp 2 đã giải quyết khá tốt vấn đề về việc khởi tạo những tham số cần thiết cho việc thực hiện hiệu ứng. Cho nên chúng em đã quyết định chọn giải pháp này.

Như vậy, với giải pháp này, khi một control mới được thêm vào, control đó sẽ được trải qua hai giai đoạn để có thể đặt hiệu ứng lên trên control đó. Hai giai đoạn này được thể hiện cụ thể ở Hình 3‑8.



Hình ‑ Quá trình đóng gói control để có thể đặt được hiệu ứng lên control đó

* ***Giai đoạn 1*** là giai đoạn đóng gói control (hay có thể gọi là bao bọc control) nhằm được control về dạng kế thừa từ BasicControl mà hệ thống có thể hiểu và quản lý.
* ***Giai đoạn 2*** là giai đoạn hiệu ứng hóa control, đây là giai đoạn làm cho BasicControl có thể đặt được các hiệu ứng lên đó. Giai đoạn này sẽ được thực hiện một cách tự động khi BasicControl được tạo ra trong công cụ.

### Thay đổi hiệu ứng cho control dạng BasicListControl

* Vấn đề

Tương tự như hiệu ứng cho BasicControl, hiệu ứng cho control dạng list cũng cần được cài đặt theo hướng tiếp cận sao cho có thể áp dụng cho control list bất kì, không phụ thuộc vào kiểu item trong list, đồng thời phải có khả năng chuyển đổi từ dạng effect này sang dạng effect mà không ảnh hưởng đến nội dung item trong list.

* Giải pháp

Để giải quyết vấn đề effect không phụ thuộc vào kiểu item trong list thì effect cần thao tác trên đối tượng tổng quát. Còn vấn đề chuyển đổi từ dạng effect này sang dạng effect khác thì effect phải có khả năng attach và detach. Lúc cần chuyển effect thì detach effect cũ và attach effect mới. Khi detach khỏi list control, effect phải đảm bảo trả list về đúng dạng ban đầu, gỡ bỏ tất cả những thay đổi khỏi list và item trong list.

Ngoài ra, khi thêm hoặc gỡ bỏ item của list sẽ ảnh hưởng đến effect nên effect cần phải biết để cập nhật lại. Cách giải quyết là không cho thao tác trực tiếp trên mảng item, mà sẽ gọi hàm của list.

### Gán hiệu ứng cho control lúc runtime

* Vấn đề

Do một control có thể có đưa ra nhiều con trỏ hiệu ứng để cho người dùng điều chỉnh hiệu ứng cho chính control đó hoặc hiệu ứng cho các control con của control đó, làm sao hiệu ứng biết cần áp đặt hiệu ứng đó cho control nào (chính control đó hoặc là control con bên trong control đó). Vấn đề là làm sao để xác định được control nào sẽ được gán hiệu ứng?

* Giải pháp

Qua việc phân tích và tìm hiểu giải pháp cho vấn đề “làm sao xác định được sẽ được gán hiệu ứng”, nhóm chúng em đã đề ra được một giải pháp để giải quyết vấn đề:

Tại lớp BasicControl sẽ cung cấp một hàm ảo có tác dụng dùng để gán hiệu ứng cho một trong những con trỏ hiệu ứng control cho phép thay đổi:

ChangeEffect(string propertyName, Type effectType, EffectableControl owner)

Với các tham số có ý nghĩa:

* propertyName: string chứa tên của con trỏ hiệu ứng cần gán hiệu ứng (ví dụ: MainEffect – con trỏ hiệu ứng mặc định có sẵn trong BasicControl)
* effectType: kiểu dữ liệu của hiệu ứng cần gán vào con trỏ hiệu ứng
* owner: là instance của EffectableControl chứa BasicControl đang xét, do hiệu ứng BasicEffect chỉ nhận tham số đầu vào là EffectableControl chứa BasicControl cần gán hiệu ứng nên cần có tham số này.

Mặc định hàm ChangeEffect chỉ có thực hiện việc khởi gán cho MainEffect, vì tại BasicControl chỉ có con trỏ hiệu ứng MainEffect.

Quá trình gán hiệu ứng cho một con trỏ hiệu ứng cần qua các bước sau:

* Bước 1: gọi hàm DetachEffect của hiệu ứng đang được con trỏ hiệu ứng giữ để hiệu ứng này trả lại trạng thái ban đầu trước khi áp hiệu ứng cho control
* Bước 2: lấy hàm dựng của hiệu ứng cần gán từ effectType của hiệu ứng. Ví du:

ConstructorInfo ci = effectType.GetConstructor(new Type[] { typeof(EffectableControl) });

* Bước 3: gọi thực thi hàm dựng này để tạo ra hiệu ứng đó. Rồi gán vào cho con trỏ hiệu ứng tương ứng. Ví dụ:

mainEffect = (BasicEffect)ci.Invoke(new object[] { owner });

Tại các control kế thừa từ BasicControl sẽ override lại hàm ChangeEffect này (nếu cần) để khởi gán hiệu ứng cho control cần gán hiệu ứng.

## List và nguồn dữ liệu

### Giao diện để tùy chỉnh list control (list editor)

* Vấn đề

Với list control, người dùng cần phải được cung cấp cách thức thực hiện các thao tác như thêm item, bỏ item, thay đổi nội dung, thứ tự item,… *PropertyGrid* (mục 3.3.2) không thể đáp ứng nhu cầu này.

* Giải pháp

Nhóm chúng em nhận thấy có hai hướng tiếp cận cho vấn đề này tương ứng với hai giải pháp:

* Giải pháp 1

Sẽ có một cách tùy chỉnh chung cho list control, không cần quan tâm list đó dùng để làm gì, chứa item gì. Với giải pháp này, việc chọn lựa thao tác và giao diện người dùng sao cho đáp ứng được nhiều loại list control rất phức tạp.

* Ưu điểm
* Có được giao diện thao tác thống nhất cho các loại list control.
* Khuyết điểm
* Cài đặt phức tạp.
* Gây khó khăn cho người sử dụng khi thao tác.
* Giải pháp 2

Với mỗi loại list control sẽ có cách thức tùy chỉnh riêng. Khi đó, editor chỉ phục vụ cho một loại list cụ thể, với những thao tác cụ thể. Như vậy thì việc cài đặt dễ dàng hơn và người sử dụng cũng thao tác tốt hơn vì giao diện editor trực quan, cung cấp đúng những chức năng cần cho loại list đó.

* Ưu điểm
* Cài đặt dễ dàng hơn.
* Tiện dụng hơn.
* Dễ dàng mở rộng khi có thêm dạng list control cần những thao tác mới.
* Khuyết điểm
* Phải có nhiều loại editor cho các dạng list khác nhau.
* Có khả năng phải cài đặt lại những thao tác giống nhau ở những editor khác nhau.

🗒 Nhận xét :

Cả hai giải pháp đều có ưu khuyết điểm riêng. Tuy nhiên, nhóm chúng em nhận thấy giải pháp 2 mang tính linh hoạt hơn, trực quan hơn giải pháp 1. Với giải pháp 2, mỗi editor chỉ phục vụ cho một dạng list control với những thao tác cụ thể nên ta có khả năng tạo ra giao diện thao tác trực quan hơn, thân thiện hơn đối với người dùng, tránh được trường hợp giao diện khó dùng vì có quá nhiều chức năng. Hơn nữa, các editor này độc lập nhau, khi gặp một dạng list mới với những thao tác mới thì chỉ cần tạo ra những editor mới tương ứng. Do đó, nhóm chúng em đã chọn giải pháp 2.

### List control và nguồn dữ liệu

* Vấn đề

Bên cạnh việc trực tiếp tạo ra dữ liệu cho list control thì người dùng còn có nhu cầu dùng list control để hiển thị dữ liệu từ một nguồn có sẵn nào đó như database, xml… Cấu trúc các nguồn dữ liệu này, mặc dù cùng là một mảng các phần tử (item) của list nhưng rất đa dạng. Ví dụ số lượng các trường dữ liệu cho item khác nhau. Cùng số trường dữ liệu nhưng thứ tự sắp xếp khác nhau. Ngoài ra, còn nhu cầu hiển thị theo những cách khác nhau cho cùng một nguồn dữ liệu hoặc chỉ hiển thị một số trường của nguồn dữ liệu. Nếu ta phân loại nguồn dữ liệu theo số trường, kiểu dữ liệu của trường… và tạo ra các kiểu list cho từng loại thì số lượng list cần có phải rất lớn!

* Giải pháp

Nhóm chúng em tìm ra cách giải quyết vấn đề này bằng cách tách riêng giao diện và dữ liệu, rồi có thêm bước kết buộc (binding) giao diện và dữ liệu. Cụ thể như sau:

* Phần dữ liệu: dữ liệu từ các nguồn khác nhau như: SQL, MySQL, XML… sẽ được lấy về và chuyển thành dạng List<List<string>>, tức là mỗi item của list control sẽ có dữ liệu là một mảng string.
* Phần giao diện: giao diện của item (item template) sẽ được thiết kế riêng, chỉ cần cung cấp các thuộc tính nhận dữ liệu kiểu string ra ngoài, tức là ta có một mảng thuộc tính nhận giá trị string để hiển thị.
* Sau đó, người dùng sẽ chọn cách kết buộc dữ liệu (một mảng string) và giao diện (một mảng thuộc tính nhận giá trị string). Ví dụ: phần tử thứ nhất mảng thuộc tính của item gắn với phần tử thứ ba mảng string (1 – 3), (2 – 5), (3 – 0), (4 – 3) … Số phần tử của mảng string và mảng thuộc tính không bắt buộc phải bằng nhau, vì có thuộc tình sẽ không được dùng (3 – 0) và có nhiều thuộc tính dùng cùng giá trị (1 – 3), (4 – 3).

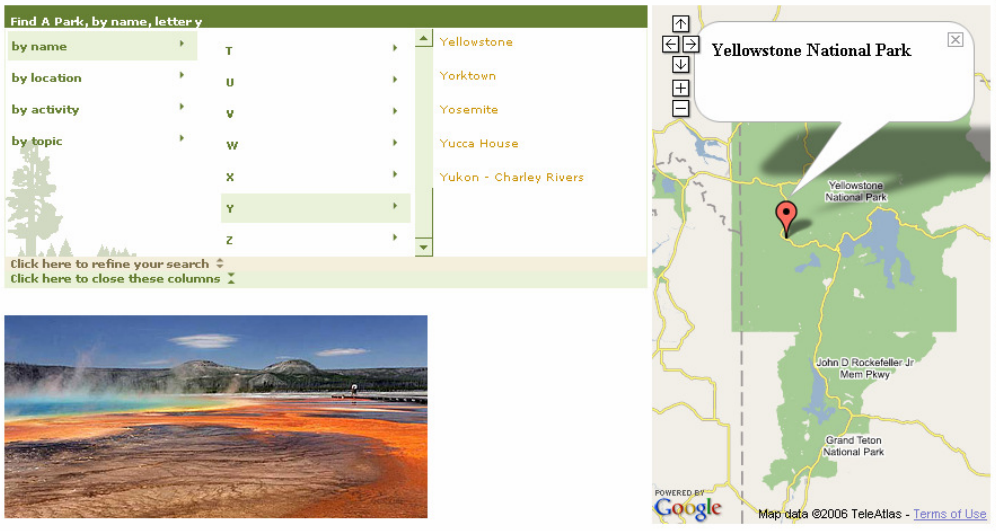
Ngoài ra, nếu muốn thêm một kiểu hiển thị khác thì chỉ cần thiết kế phần giao diện rồi thêm vào chương trình, không cần thực hiện các xử lý liên quan đến dữ liệu vì đã được chương trình hỗ trợ.

## Phối hợp xử lý giữa các control

### Liên kết sự kiện giữa các control

* Vấn đề

Một control có thể sẽ có nhu cầu phát sinh ra những sự kiện để các control khác có thể xử lý, và control cũng có thể có nhu cầu kích hoạt một xử lý gì đó dựa vào sự kiện của control khác phát sinh. Hình 3‑9 là một ví dụ về việc phối hợp xử lý giữa các control.



Hình ‑ Ví dụ minh họa về phối hợp xử lý giữa các control [5]

Trong ví dụ, khi người dùng chọn một địa danh từ danh sách, control bên dưới sẽ hiển thị những hình ảnh liên quan đến địa danh đó mà control đó tìm được trên dịch vụ tìm hình ảnh của Flickr, đồng thời bản đồ bên cạnh sẽ hiển thị vị trí tương ứng của địa danh đó.

Vấn đề được đặt ra là cần quản lý những thông tin gì và cần thực hiện những thao tác nào, làm sao để quản lý các sự kiện và hàm xử lý, khi nào cần bắt sự kiện, khi nào cần gọi hàm xử lý?

* Giải pháp

Qua phân tích vấn đề tương tác giữa các control, nhóm xác định được cần giải quyết các vấn đề sau:

* Làm sao biết và quản lý các sự kiện của control (mục 3.6.2)
* Làm sao biết và quản lý các hàm xử lý của control (mục 3.6.2)
* Khi nào cần bắt sự kiện (mục 3.6.3)
* Khi nào cần gọi hàm xử lý (mục 3.6.3)

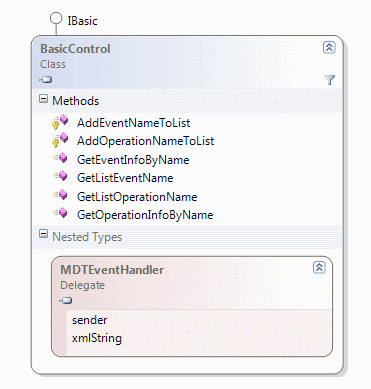
### Cơ chế khai báo

Các sự kiện có thể được phát sinh với số lượng tham số khác nhau và kiểu dữ liệu của tham số cũng khác nhau. Trong khi đó, muốn xử lý một sự kiện nào đó thì hàm xử lý phải biết được số lượng tham số và kiểu dữ liệu của tham số là gì. Nếu làm theo cách này, một control muốn cung cấp hàm xử lý để thực hiện một chức năng nào đó thì phải cung cấp nhiều hàm xử lý với danh sách tham số khác nhau để xử lý cho những sự kiện khác nhau. Hơn nữa, các loại sự kiện với kiểu và số lượng tham số khác nhau là rất đa dạng, không thể biết trước được. Vì thế, giải pháp này là không khả thi.

Như thế, cần quy định ra một cơ chế khai báo cho sự kiện và hàm xử lý để có quản lý được. Nhóm chúng em đã quy định ra cơ chế khai báo có thể áp dụng tổng quát như sau:

* Đối với sự kiện (event): nhóm đã đặt ra một loại sự kiện với tham số truyền ra của loại sự kiện gồm hai tham số, tham số thứ nhất là đối tượng đã phát sinh ra sự kiện và tham số thứ hai là chuỗi xml chứa những thông tin mà đối tượng phát sinh muốn truyền kèm theo khi sự kiện được phát sinh. Lý do quy định truyền ra chuỗi là vì đây là loại dữ liệu mà mọi hệ máy đều có thể hiểu và xử lý được.
* Đối với hàm xử lý (operation): nhóm quy định chỉ gọi những hàm xử lý nhận vào một tham số là chuỗi xml. Hàm xử lý này sẽ phải phân tích chuỗi xml được truyền vào để lấy dữ liệu hàm đó cần.
* Ngoài ra, nhóm chúng em còn quy định những sự kiện và hàm xử lý này cần phải được đăng ký với hệ thống công cụ để hệ thống có thể biết và kiểm tra chắc chắn là đúng quy định về loại sự kiện và hàm xử lý mà nhóm đã đặt ra.

Hình 3‑10 là sơ đồ lớp mà nhóm đã cài đặt giải pháp vừa được nêu trên Silverlight. *MDTEventHanlder* chính là delegate quy định kiểu sự kiện; các hàm *AddEventNameToList* và *AddOperationNameToList* là hai hàm để đăng ký sự kiện và hàm xử lý với hệ thống. Ngoài ra, nhóm còn cung cấp các hàm lấy danh sách sự kiện (*GetListEventName*) và hàm xử lý (*GetListOperationName*) để các thành phần bên ngoài control có thể lấy được.



Hình ‑ Các hàm và delegate trong BasicControl liên quan đến việc liên kết sự kiện

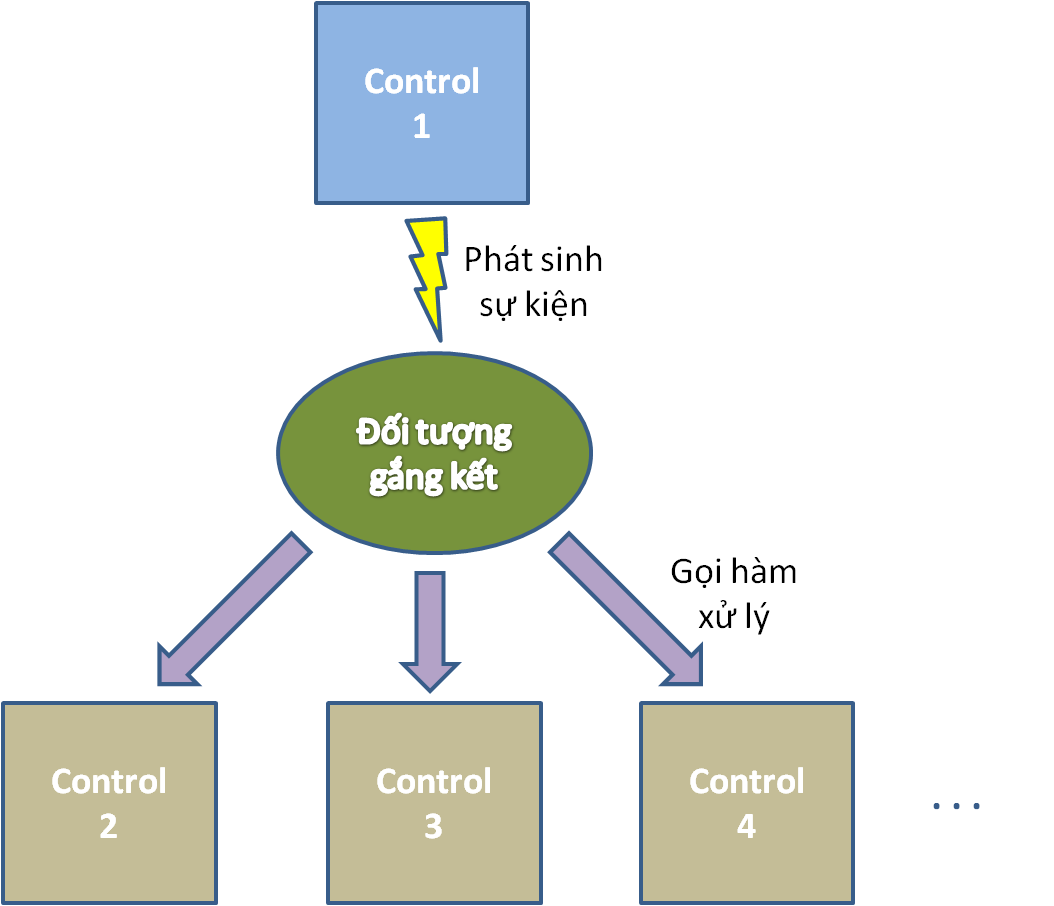
### Cơ chế lắng nghe và gắng kết sự kiện

Qua xem xét, nhóm nhận thấy một liên kết sự kiện cần biết được những thông tin sau :

* Đối tượng phát sinh sự kiện
* Định danh (tên) của sự kiện phát sinh
* Những đối tượng cần xử lý sự kiện và hàm xử lý tương ứng

Do đó, để lắng nghe một sự kiện thì cần biết đối tượng chứa sự kiện và định danh của sự kiện cần lắng nghe. Thêm vào đó, để gắng kết sự kiện thì trong cần biết được những đối tượng có nhu cầu xử lý sự kiện phát sinh và hàm xử lý sẽ được gọi từ những đối tượng đó.

Nhóm đã không trực tiếp chỉ định khi một sự kiện được phát sinh sẽ phải gọi những hàm xử lý nào. Nhóm đã tiến hành lắng nghe một sự kiện bởi một hàm xử lý duy nhất. Khi sự kiện được phát sinh thì hàm xử lý này sẽ lần lượt phân phối sự kiện bằng cách gọi hàm xử lý của những đối tượng có nhu cầu xử lý sự kiện đó. Quá trình này được mô tả ở Hình 3‑11.



Hình ‑ Quá trình lắng nghe và gắng kết sự kiện

### Quản lý những sự kiện được liên kết

Nhóm nhận thấy rằng không phải sự kiện nào cũng cần phải lắng nghe, chỉ những sự kiện được người dùng chọn để xử lý mới cần được lắng nghe. Do đó, hệ thống cần quản lý để biết những sự kiện đã được đăng ký.

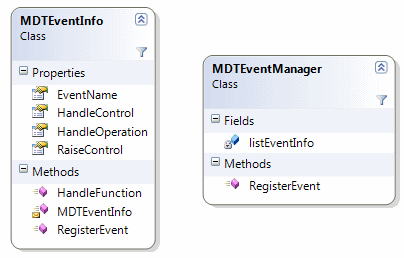
Chính vì thế, nhóm đã thiết kế và cài đặt một lớp để quản lý, lớp này sẽ lưu trữ danh sách các sự kiện được lắng nghe. Đồng thời, lớp này cung cấp một hàm để đăng ký sự kiện, hàm này sẽ thực hiện việc kiểm tra trước đó đã sự kiện đã được lắng nghe chưa, nếu có sẽ gỡ bỏ việc lắng nghe này, sau đó sẽ tiến hành thao tác lắng nghe lại sự kiện. Ý tưởng để cài đặt lớp này không phục thuộc vào công nghệ sử dụng.

Nhóm đã sử dụng Silverlight để cái đặt lớp này với tên *MDTEventManager*. Trong đó, cung cấp *RegisterEvent* để đăng ký lắng nghe sự kiện.

### Vấn đề triển khai trên Silverlight

Việc cài đặt việc phối hợp xử lý giữa các sự kiện trên Silverlight theo giải pháp đã nêu khá đơn giản. Cùng với việc thêm vào *BasicControl* các hàm và một delegate cho sự kiện như mô tả ở Hình 3‑10, nhóm còn cài đặt thêm hai lớp như sau:

* Lớp *MDTEventInfo* chứa thông tin về một liên kết sự kiện bao gồm đối tượng phát sinh sự kiện, tên sự kiện, các đối tượng và hàm xử lý tương ứng cần gọi khi sự kiện được phát sinh. Nhiệm vụ của lớp này là để quản lý các thông tin liên quan đến một sự kiện, lắng nghe và gắng kết sự kiện (cơ chế đã nêu ở mục 3.6.3).
* Lớp *MDTEventManager* chứa danh sách các sự kiện đã được đăng ký gắng kết (danh sách các đối tượng *MDTEventInfo*). Và cung cấp hàm *RegisterEvent* để đăng ký gắng kết sự kiện từ bên ngoài.



Hình ‑ Lớp MDTEventInfo và MDTEventManager

Trong quá trình cài đặt, chúng em đã gặp phải vấn đề là làm sao để có thể đăng ký và gỡ bỏ sự kiện động lúc runtime. Vấn đề này được nhóm giải quyết bằng sử dụng kỹ thuật Reflection được hỗ trợ trong .NET. Sau đây là giới thiệu sơ lược về cách sử dụng kỹ thuật Reflection để đăng ký và gỡ bỏ sự kiện:

* Đầu tiên cần lấy được đối tượng *EventInfo* của sự kiện cần bắt. Để lấy được đối tượng EventInfo, cần thực hiện các bước:
* Lấy đối tượng *Type* của đối tượng cần bắt sự kiện bằng cách gọi hàm *GetType* của đối tượng đó.
* Gọi hàm *GetEvent* với tham số là tên của sự kiện cần lấy EventInfo. Hàm này sẽ trả về đối tượng EventInfo cần lấy.

EventInfo ei = control.GetType().GetEvent("ten su kien muon lay");

* Để đăng ký sự kiện, gọi hàm *AddEventHandler* của đối tượng EventInfo với tham số thứ nhất là đối tượng sẽ phát sinh sự kiện, tham số hai là đối tượng delegate để xử lý sự kiện. Kiểu dữ liệu của delegate đó được cho biết thông qua property EventHandlerType của đối tượng EventInfo.

Type delegateType = ei.EventHandlerType;

ei.AddEventHandler(control, Delegate.CreateDelegate(delegateType, obj, "Ten ham xu ly su kien")); // voi obj la doi tuong chua ham xu ly su kien co ten tuong ung voi tham so thu 3

* Để gỡ bỏ sự kiện, gọi hàm *RemoveEventHandler* của đối tượng EventInfo với các tham số tương tự như tham số của hàm AddEventHandler.

# Một số vấn đề và giải pháp về khả năng mở rộng, sử dụng và triển khai

*✍ Nội dung chương này sẽ trình bày về các vấn đề cũng như giải pháp cho khả năng mở rộng hệ thống công cụ thiết kế, sử dụng ứng dụng đã được thiết kế và những vấn đề riêng khi triển khai trên Silverlight 4.*

## Khả năng mở rộng hệ thống

### Load động một control hoặc hiệu ứng bất kỳ vào chương trình

* Vấn đề

Làm thế nào để ta có thể load động một control hoặc hiệu ứng bất kỳ trong chương trình một cách dễ dàng. Nói cách khác là công cụ sẽ cho phép thêm control hoặc hiệu ứng mới vào lúc runtime. Vấn đề được đặt ra là công cụ cần hỗ trợ việc thêm vào các control hoặc hiệu ứng mà không cần phải biên dịch lại chương trình bằng tay?

* Giải pháp

Qua việc tìm hiểu các công nghệ hỗ trợ thì nhóm đã đề xuất ra được hai giải pháp:

* Giải pháp 1

Lưu trữ các control và hiệu ứng dưới dạng các file \*.cs và \*.xaml mà không biên dịch trước. Lúc runtime sẽ tiến hành biên dịch control hoặc hiệu ứng cần dùng rồi tạo ra đối tượng.

* Ưu điểm
* Không cần biên dịch trước các control hoặc hiệu ứng.
* Khuyết điểm
* Việc biên dịch control hoặc hiệu ứng luôn được thực hiện trong những lần sử dụng khác nhau.
* Nếu trong control hoặc hiệu ứng có tham chiếu đến một dll nào đó không có trong chương trình sẽ gây ra lỗi. Và việc xác định được dll tham chiếu tương đối phức tạp.
* Giải pháp 2

Tiến hành biên dịch control hoặc hiệu ứng thành tập tin dll. Sau đó sẽ lưu trữ lại dll này đồng thời cũng lưu trữ thêm các dll mà được dll này tham chiếu đến. Lúc runtime sẽ dùng kỹ thuật reflection trong .NET để load tất cả các dll liên quan sau đó sẽ tạo ra đối tượng cần thiết.

* Ưu điểm
* Chỉ cần biên dịch control hoặc hiệu ứng một lần để tạo ra tập tin dll.
* Giải quyết được vấn đề dll tham chiếu đến bằng cách chép theo những dll tham chiếu đó đi cùng với dll chính.
* Khuyết điểm
* Sẽ có thể xảy ra hiện tượng hai dll cùng tên.

🗒 Nhận xét

Qua việc khảo sát tính khả thi của hai giải pháp trên. Nhóm đã quyết định chọn giải pháp 2 do giải pháp này có thể giải quyết được vấn đề dll tham chiếu tương đối đơn giản.

### Xác định các control và hiệu ứng được chương trình hỗ trợ

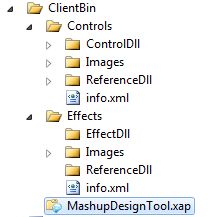
* Vấn đề

Chương trình hoạt động trên nền web, do đó tất cả các control và hiệu ứng của chương trình đều được lưu trữ trên server và khi cần thì client sẽ tải về những control và hiệu ứng này. Làm thế nào để client có thể biết được hiện chương trình đã hỗ trợ bao nhiêu loại control, bao nhiêu loại hiệu ứng. Điều đó có nghĩa là xác định hiện trên server đang có bao nhiêu control, bao nhiêu hiệu ứng, và làm sao biết được dll dùng để tạo ra control hoặc hiệu ứng nào, và dll đó đã tham chiếu đến những dll nào. Vấn đề đặt ra là cần đưa ra một cấu trúc lưu trữ để có thể cho biết tác dụng của dll bất kỳ là gì?

* Giải pháp

Qua việc tìm hiểu “đưa ra một cấu trúc lưu trữ để biết được tác dụng của một dll” thì chúng em đã đề ra được một cấu trúc lưu trữ như sau:

* ***Cấu trúc thư mục***: đây là cấu trúc thư mục để lưu trữ các dll và các tập tin liên quan. Cấu trúc thư mục được thể hiện ở



Hình ‑ Cấu trúc thư mục lưu trữ dll và các tập tin liên quan

Trong đó, cấu trúc thư mục bao gồm:

* Thư mục Controls, Effects: là hai thư mục nằm cùng cấp với tập tin ứng dụng .xap.
* Thư mục ControlDll, EffectDll: là hai thư mục để chứa các tập tin dll của control hoặc hiệu ứng.
* Thư mục ReferenceDll: là thư mục chứa tất cả các tập tin dll được tham chiếu đến từ dll của control hoặc hiệu ứng.
* Thư mục Images: là thư mục chứa hình ảnh đại diện cho control hoặc hiệu ứng.
* Tập tin info.xml: là tập tin mô tả thông tin.
* ***Tập tin info.xml***: đây là tập tin thông tin về các control hoặc hiệu ứng đang có. Với mỗi control hoặc hiệu ứng, tập tin sẽ lưu trữ các thông tin:
* Tên lớp của control hoặc hiệu ứng (Name)
* Tên hiển thị của control hoặc hiệu ứng (DisplayName)
* Tên tập tin dll chứa control hoặc hiệu ứng (DllFilename)
* Mô tả về control hoặc hiệu ứng (Description)
* Danh sách dll tham chiếu đến (DllReferences)
* Hình ảnh đại diện (IconName)
* Nhóm của control hoặc hiệu ứng (Group)
* Và một vài thông tin khác (nếu có)

Xml schema của info.xml như sau:

* Tập tin info.xml trong thư mục Controls:

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Controls">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Control">

<xs:complexType>

<xs:sequence>

<xs:element name="Name" type="xs:string" />

<xs:element name="DllFilename" type="xs:string" />

<xs:element name="DllReferences">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="DisplayName" type="xs:string" />

<xs:element name="Description" type="xs:string" />

<xs:element name="IconName" type="xs:string" />

<xs:element name="Group" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

* Tập tin info.xml trong thư mục Effects:

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Effects">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Effect">

<xs:complexType>

<xs:sequence>

<xs:element name="Name" type="xs:string" />

<xs:element name="DllFilename" type="xs:string" />

<xs:element name="DllReferences">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="DisplayName" type="xs:string" />

<xs:element name="Description" type="xs:string" />

<xs:element name="IconName" type="xs:string" />

<xs:element name="Group" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

* Ưu điểm
* Xác định được đâu là dll để tạo control hoặc hiệu ứng, đâu là dll được tham chiếu đến. Đồng thời cũng xác định một dll control hoặc hiệu ứng sẽ tham chiếu đến những dll nào.
* Khi client gửi yêu cầu để lấy danh sách các control hoặc hiệu ứng đang có thì server đơn giản chỉ việc trả về cho client tập tin mô tả info.xml mà không cần phải có những thao tác xử lý phức tạp.
* Khuyết điểm
* Khi cần thêm một control hoặc hiệu ứng mới thì cần phải tuân theo cấu trúc thư mục này. Đồng thời phải cập nhật lại vào tập tin info.xml. Nếu quên cập nhật tập tin này thì xem như không có control hoặc hiệu ứng đó trong chương trình.
* Nếu thông tin về tên dll và tên của những dll được tham chiếu đến trong tập tin info.xml bị sai sẽ thì control hoặc hiệu ứng đó sẽ không thể load được.

### Tạo ra đối tượng control hoặc hiệu ứng

* Vấn đề

Để tạo được một đối tượng, chương trình cần phải có đầy đủ các dll cần thiết để tạo ra đối tượng này. Những dll cần thiết là dll chứa đối tượng đó và những dll được dll chứa đối tượng có tham chiếu (reference) tới. Do đó, để tạo được một đối tượng control hoặc hiệu ứng, cần chương trình cần phải có được những dll cần thiết liên quan tới control hoặc hiệu ứng đó. Thông tin về những dll này được nhóm quy định là lưu trữ trong tập tin info.xml, cấu trúc và nội dung của tập tin này được mô tả tại mục 4.1.2. Vậy làm sao để có thể tạo ra được đối tượng từ những thông tin về tên dll chứa control hoặc hiệu ứng và tên những dll được reference tới. Nghĩa là làm sao để có được các dll đó tại client và làm sao để tạo ra được đối tượng khi đã có đủ các dll? Vấn đề đặt ra là làm sao để lấy được những dll cần thiết về client để tạo được đối tượng?

* Giải pháp

Qua việc phân tích vấn đề và tìm hiểu sự hỗ trợ của công nghệ .NET trong Silverlight, chúng em đã đề ra giải pháp gồm hai bước:

* ***Bước 1***: tải tất cả các dll cần thiết cho việc tạo control hoặc hiệu ứng (dll chứa control hoặc hiệu ứng và các dll được tham chiếu đến) về client. Để thực hiện bước này, chúng em đã sử dụng hàm *OpenReadAsync* của đối tượng *WebClient* trong namespace System.Net của .NET với tham số là đường dẫn URL đến tập tin cần tải về, đồng thời bắt sự kiện *OpenReadCompleted*.

Uri uri = new Uri(filePath, UriKind.Absolute);

WebClient webClient = new WebClient();

webClient.OpenReadCompleted += new OpenReadCompletedEventHandler(webClient\_DownloadControlCompleted);

webClient.OpenReadAsync(uri);

* ***Bước 2***: load các dll đã tải về vào chương trình. Để thực hiện bước này, nhóm đã sử dụng kỹ thuật Reflection được .NET hỗ trợ. Trong hàm xử lý sự kiện *OpenReadCompleted*, chúng em dùng hàm *Load* trong đối tượng *AssemblyPart* để load dữ liệu được tải về vào chương trình. Một khi đã tải về đầy đủ tất cả các dll cần thiết thì chúng em sẽ dùng hàm *CreateInstance* trong đối tượng *Assembly* với tham số là tên lớp của đối tượng cần tạo.

private void webClient\_DownloadControlCompleted(object sender, OpenReadCompletedEventArgs e)

{

try

{

if (e.Error == null)

{

AssemblyPart assemblyPart = new AssemblyPart();

Assembly assembly = assemblyPart.Load(e.Result);

FrameworkElement uc = (FrameworkElement) LoadedAssembly[controlName].CreateInstance(controlName);

}

}

catch { }

}

* ***Ngoài ra*,** nhóm còn tiến hành đánh dấu lại những tập tin dll nào đã được tải về client để khi một control khác cần dùng đến dll đã được tải về trước đó sẽ không cần tải lại nữa.

## Vấn đề ghi nhận và trình diện lại ứng dụng được thiết kế

### Ghi nhận và trình diễn lại những control, hiệu ứng và điều chỉnh của người dùng

* Vấn đề

Một trong những yêu cầu hiển nhiên đó với chương trình là cần phải ghi nhận lại những control, hiệu ứng và những điều chỉnh của người dùng trên giao diện thiết kế, sau đó phải trình diễn lại một cách chính xác ứng dụng người dùng đã thiết kế. Ngoài những thông số (là những giá trị của thuộc tính) của các control và hiệu ứng, cần lưu trữ những gì nữa để cho công việc tái tạo được chính xác và dễ dàng. Vấn đề được đặt ra là cần xác định sẽ lưu trữ những thông tin gì và thứ tự thực hiện của việc lưu trữ như thế nào, đồng thời cũng cần xác định các bước và cách thức thực hiện ở giai đoạn tái tạo?

* Giải pháp

Qua việc phân tích những thông tin liên quan đến ứng dụng được người dùng thiết kế, nhóm đã xác định được những thông tin cần lưu trữ là:

* Các thông tin về thông số của control và hiệu ứng
* Thông tin về các sự kiện được bắt để xử lý
* Thông tin về những tập tin dll cần dùng
* Một số thông tin về Canvas chứa control.

Các bước thực hiện của giai đoạn lưu trữ và tái tạo:

* ***Giai đoạn lưu trữ***: lần lượt theo thứ tự:
* Xác định và lưu lại những dll cần thiết.
* Lưu lại một số thông tin về Canvas (kích thước, màu nền).
* Lưu trữ lại thông số của mỗi control và thông số của những hiệu ứng của control đó.
* Cuối cùng là lưu lại những thông tin về các sự kiện được xử lý.
* ***Giai đoạn tái tạo***: lần lượt theo thứ tự:
* Xác định những dll cần tải về chương trình và tải những dll này về.
* Sau khi tải đã tải tất cả các dll về, tiến hành tạo ra Canvas và gán lại những giá trị cần thiết.
* Tiến hành tạo control và hiệu ứng trên control đó.
* Cuối cùng là tiến hành đăng ký lại tất cả các sự kiện đã lưu.

### Save và load một đối tượng

* Vấn đề

Việc save/load control, hiệu ứng và một số thông tin khác đã đươc đề cập có thể được xem xét là việc save/load các thuộc tính của một đối tượng nào đó. Ngoài ra, do Silverlight không hỗ trợ sẵn hàm Clone control nên chúng em cũng dùng luôn hàm save/load cho việc Clone control. Vấn đề đặt ra là làm sao để có thể save/load đối tượng đó.

* Giải pháp

Ứng cử việc đầu tiên được nhóm tìm hiểu để giải quyết vấn đề đó là lớp *Xmlserializer* được hỗ trợ trong Silverlight. Đây là lớp để save/load theo định dạng xml của đối tượng được đưa vào. Nhưng nếu trong đối tượng đó chứa những thuộc tính có kiểu dữ liệu *Dictionary* thì sẽ gặp lỗi vì lớp XmlSerializer không hỗ trợ kiểu dữ liệu này. Do bên trong control của Silverlight có một số thuộc tính có kiểu dữ liệu là Dictionary nên không sử dụng lớp XmlSerializer này.

Vì thế, với mục đích tìm cách ra giải pháp để có thể save lại control, nhóm đã tìm hiểu được trong Silverlight có hàm *XamlReader* dùng để load một nội dung xaml. Nhưng tiếc thay Silverlight lại không hỗ trợ hàm save ra xaml. Trên Internet cũng có một số cài đặt để thực hiện việc save control ra dạng xaml, nhưng tất cả đều ở dạng chưa ổn định, còn gặp nhiều lỗi.

Chính vì thế, nhóm chúng em quyết định tự cài đặt lớp *MyXmlSerializer* dùng cho việc save/load đối tượng trong chương trình. Việc cài đặt lớp này sao cho đảm bảo tính ổn định là khá khó khăn nhưng được ưu điểm là có thể chọn lựa những cái cần save và cấu trúc của xml.

* **Cách thực hiện:**

*Cách lấy và gán giá trị cho một thuộc tính (property) của đối tượng bằng kỹ thuật reflection:* (xem mục 3.3.2)

*Khi tiến hành cài đặt thì nhóm em nhận thấy đối tượng cần save được phân thành 2 loại:*

* Loại 1: đối tượng thông thường, dữ liệu cần save chính là các thuộc tính của đối tượng.
* Loại 2: đối tượng đặt biệt, cụ thể là dạng list, collection, và Array, bỏ qua kiểu Dictionary, Array chỉ xét kiểu array một chiều. Với loại này thì dữ liệu cần lưu không phải là thuộc tính của đối tượng mà chính là các phần tử của đối tượng, các phần tử này được truy xuất theo dạng indexer (dùng dấu []).

*Cách xác định đối tượng thuộc loại nào:*

* Bước 1: lấy Type của đối tượng bằng hàm GetType.
* Bước 2: Với đối tượng Type lấy được, dùng hàm *IsAssignableFrom* để kiểm tra đối tượng có thực thi từ interface *IList* hay không, nếu có thì nó thuộc loại 2. Xét thuộc tính IsArray trong đối tượng Type để xác định nó là kiểu Array. Vì cách lấy dữ liệu của Array khác với cách lấy dữ liệu của list hoặc collection.

*Hàm save đối tượng loại 1: hàm Save(object obj)*

* Bước 1: nếu đối tượng là kiểu cơ sở (IsPrimitive = true) thì ta sẽ ghi giá trị, kết thúc hàm. Ở bước này ta cũng xét thêm một số trường hợp đặc biệt là kiểu *string*, *Enum* mặc dù nó không phải kiểu cơ sở nhưng ta sẽ ghi giá trị của nó là giá trị obj.ToString().
* Bước 2: ta sẽ duyệt qua hết các thuộc tính có trong đối tượng. Lấy giá trị của từng thuộc tính, kiểm tra thuộc tính đó là đối tượng thuộc loại nào, loại 1 thì gọi hàm Save, loại 2 thì gọi hàm SaveList hoặc SaveArray.

*Hàm save đối tượng loại 2: hàm SaveList(object obj) và SaveArray(object obj), 2 hàm này cách làm việc giống nhau, chỉ khác nhau ở chi tiết kỹ thuật.*

* Bước 1: Lấy số lượng phần tử. Với đối tượng là *Array* thì ép kiểu về Array rồi lấy giá trị *Length*. Với đối tượng là *List* hoặc *Collection* thì lấy giá trị thuộc tính *Count*.
* Bước 2: Lấy giá trị từng phần tử, xét phần tử đó là đối tượng thuộc loại 1 hoặc loại 2 rồi gọi hàm tương ứng.
  + Với đối tượng kiểu Array thì lấy giá trị phần tử bằng hàm GetValue(i) của đối tượng Array.
  + Với đối tượng là list hoặc collection, đầu tiên là lấy ProperytInfo của thuộc tính Item bằng hàm GetValue(“Item”). Sau đó lấy giá trị phần tử bằng hàm GetValue(obj,new int[]{i}) từ đối tượng PropertyInfo vừa lấy.

*Cấu trúc của chuỗi xml:*

* Đối tượng loại 1:

<Root Type="kieu cua doi tuong">

<TenThuocTinh Type="kieu cua doi tuong">

Gia tri (co the la doi tuong loai 1 hoac loai 2).

</TenThuocTinh>

<TenThuocTinh Type="kieu cua doi tuong">

Gia tri (co the la doi tuong loai 1 hoac loai 2).

</TenThuocTinh >

…

</Root>

* Đối tượng loại 2:

<Root Type="kieu cua doi tuong">

<Child Type=" kieu cua doi tuong ">

Gia tri (co the la doi tuong loai 1 hoac loai 2).

</Child>

<Child Type=" kieu cua doi tuong ">

Gia tri (co the la doi tuong loai 1 hoac loai 2).

</Child>

…

</Root>

*Tương tự khi save, việc load cũng được chia làm loại dựa theo loại đối tượng:*

Để xác định được là đang load loại đối tượng nào, nhóm sẽ lấy đối tượng *Type* bằng cách gọi hàm *GetType* với tham số là chuỗi chứa trong thuộc tính Type của thẻ đang xét.

*Hàm load đối tượng loại 1: hàm Load(object obj, XElement element)*

* Bước 1: tạo ra đối tượng dựa vào kiểu dữ liệu theo chuỗi chứa trong thuộc tính Type của thẻ element.
* Bước 2: nếu đối tượng là kiểu cơ sở (IsPrimitive = true), ta sẽ tạo ra đối tượng thuộc kiểu dữ liệu đó và trả ra đối tượng vừa tạo, đồng thời kết thúc gọi đệ quy tại đây. Ngược lại sẽ qua bước 3.
* Bước 3: ta sẽ duyệt qua hết các thẻ con có trong thẻ element đang xét. Mỗi thẻ con tương ứng với một property của đối tượng vừa được tạo. Tùy theo kiểu dữ liệu của property mà sẽ gọi hàm *Load* nếu là đối tượng loại 1, hàm *LoadArray* hoặc *LoadList* nếu là đối tượng loại. Kết quả trả về của việc gọi hàm sẽ được gán vào cho property của đối tượng vừa tạo.

*Hàm load đối tượng loại 2: hàm LoadList(object obj, XElement element) và LoadArray(object obj, XElement element). Tương tự với hai hàm save SaveList và SaveArray, hai hàm này cũng cách làm việc giống nhau, chỉ khác nhau ở chi tiết kỹ thuật.*

* Bước 1: tạo ra đối tượng dựa vào kiểu dữ liệu theo chuỗi chứa trong thuộc tính Type của thẻ element.
  + Đối với List, gọi hàm *Activator.CreateInstance* với tham số là kiểu dữ liệu của list cần tạo.
  + Đối với Array, gọi hàm *Array.CreateInstance* với tham số là kiểu dữ liệu của từng phần tử trong Array và số lượng phần tử. Kiểu dữ liệu của phần tử trong Array có thể xác định bằng cách gọi hàm *GetElementType* của đối tượng Type của Array đang xét.
* Bước 2: ta sẽ duyệt qua hết tất cả các thẻ con có tên Child của thẻ element. Gọi đệ quy hàm Load để tạo ra đối tượng từ những thẻ con này. Gán đối tượng được trả ra từ việc gọi hàm vào cho Array hoặc List.
  + Đối với List, gọi hàm *Add* để thêm phần tử tạo.
  + Đối với Array, gọi hàm *SetValue(value, i)* để gán phần tử vừa tạo vào vị trí thứ i trong Array.
* Bước 3: trả ra đối tượng Array hoặc List sau khi đã load xong.

### Save và load những tập tin dll cần dùng

Thực chất, việc lưu lại những tập tin dll cần dùng là lưu lại tên đường dẫn đến những tập tin này trên server.

Trong quá trình thiết kế, người dùng có thể chọn tạo cũng như xóa nhiều control hoặc hiệu ứng khác nhau. Mỗi control hoặc hiệu ứng này có thể cần sử dụng những dll khác nhau để tạo ra chúng. Do đó phương án lưu lại tất cả các dll đã được tải về client sẽ gây ra lãng phí, vì có thể sẽ lưu lại những dll không cần dùng đến. Lúc load lại, chương trình cũng sẽ tải lại những dll không dùng tới này, điều này sẽ làm chương trình load lại chậm hơn. Vì thế, chỉ cần lưu lại những tập tin dll mà được dùng bởi một trong những control hoặc hiệu ứng cần tái tạo. Vậy làm thế nào xác định những tập tin dll nào lưu?

Nhóm đã giải quyết bằng cách duyệt từng control và hiệu ứng cần lưu lại, lấy tên của những dll liên quan đến control và hiệu ứng đó rồi đưa vào danh sách những dll cần lưu nếu tên dll này chưa có trong danh sách này. Sau khi đã duyệt hết tất cả các control và hiệu ứng cần lưu, nhóm sẽ tiến hành chuyển danh sách tập tin dll cần thiết này thành một chuỗi có cấu trúc xml. Xml schema của chuỗi xml đó như sau:

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="NeccesaryDlls">

<xs:complexType>

<xs:sequence>

<xs:element name="ControlDll">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="ControlReferenceDll">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="EffectDll">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="EffectReferenceDll">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Dll" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Khi load, chương trình sẽ dựa theo tên tập tin và thư mục chứa dll để xác định đường dẫn đến tập tin dll đó và lần lượt sử dụng đối tượng *WebClient* để tải những dll này về và load vào chương trình.

### Save và load 1 số thông tin về Canvas chứa control

Những thông tin về Canvas cần lưu là: chiều dài (*Width*), chiều cao (*Height*) và màu nền (*Background*). Mỗi thông tin sẽ được chuyển thành chuỗi xml theo nguyên tắc sau:

* Thẻ gốc sẽ có tên của property cần lưu
* Giá trị của property sẽ được serialize thành chuỗi xml là thẻ con của thẻ gốc.
* Ví dụ:

<Width>

<Double Type="System.Double, mscorlib, Version=2.0.5.0, Culture=neutral, PublicKeyToken=7cec85d7bea7798e">500</Double>

</Width>

Khi load lại, chương trình sẽ dựa vào tên của thẻ gốc để xác định thuộc tính cần gán. Chương trình sẽ tiến hành deserialize giá trị của property này dựa vào thẻ con bên trong thẻ gốc và gán giá trị đó cho property của Canvas.

### Save và load control và hiệu ứng của control

Do *BasicControl* chỉ cho phép chỉnh sửa các thuộc tính có trong danh sách được lấy ra từ hàm *GetParameterNameList* nên khi save chỉ cần save lại những property trong danh sách này mà không cần lưu lại tất cả các thuộc tính có trong control. Ngoài ra, cần lưu thêm các thông tin về *Left*, *Top*, *ZIndex* và *DockType* của *EffectableControl* chứa control này.

Với hiệu ứng của control, ta sẽ lấy danh sách các con trỏ hiệu ứng có trong control bằng hàm *GetListEffectName*. Với mỗi con trỏ hiệu ứng sẽ lấy hiệu ứng đang được giữ bởi con trỏ này. Tương tự như control, các hiệu ứng cũng chỉ cần lưu lại những property có trong danh sách được lấy từ hàm *GetParameterNameList.*

Những giá trị cần lưu đã nêu ở trên sẽ được lưu lại thành chuỗi có định dạng xml. Với các property cần lưu sẽ được chuyển thành chuỗi xml bằng cách dùng lớp *MyXmlSerializer* được mô tả ở mục 4.2.2

Cấu trúc của chuỗi xml được tạo ra:

<Control Type="kieu du lieu cua control">

<Left Type="kieu du lieu double">Canvas.Left cua EffectableControl</Left>

<Top Type="kieu du lieu double">Canvas.Top cua EffectableControl</Top>

<ZIndex Type="kieu du lieu int">Canvas.ZIndex cua EffectableControl</ZIndex>

<DockType Type="Kieu enum DockType">DockCanvas.Docktype cua EffectableControl"</DockType>

Cac thuoc tinh khac ...

<Effects>

<TenHieuUng Type="Kieu du lieu cua hieu ung">

Cac property cua hieu ung

</TenHieuUng >

<TenHieuUng Type="Kieu du lieu cua hieu ung">

Cac property cua hieu ung

</TenHieuUng >

...

</Effects>

</Control>

Khi load, cần thực hiện theo các bước sau:

* Tạo ra control với kiểu dữ liệu ứng, load tất cả các thuộc tính đã lưu (trừ Left, Top, ZIndex, DockType) và gán vào cho control.
* Tạo ra đối tượng EffectableControl chứa control rồi load các thuộc tính Left, Top, ZIndex, DockType để gán cho EffectableControl.
* Lấy tên và kiểu dữ liệu tương ứng của hiệu ứng, sau đó gọi hàm ChangeEffect của control để tạo ra đối tượng hiệu ứng. Load và gán lại giá trị các thuộc tính của hiệu ứng.

### Save và load các sự kiện đã được liên kết

Những thông tin cần lưu trữ về một liên kết sự kiện với hàm xử lý sự kiện đó là:

* Control phát sinh sự kiện
* Tên sự kiện phát sinh
* Các control có hàm xử lý sự kiện
* Tên hàm xử lý sự kiện cần gọi tương ứng với control.

Những thông tin này được chứa trong đối tượng *MDTEventInfo.*Với tên sự kiện và tên những hàm xử lý sự kiện, ta chỉ việc lưu lại chuỗi string chứa tên đó. Với control phát sinh sự kiện và các control có hàm xử lý sự kiện, nhóm chúng em chỉ lưu lại chỉ số của control trên những control đã được lưu.

XmlSchema của chuỗi xml được tạo ra:

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

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="Events">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Event">

<xs:complexType>

<xs:sequence>

<xs:element name="RaiseControlIndex" type="xs:unsignedByte" />

<xs:element name="EventName" type="xs:string" />

<xs:element name="Handles">

<xs:complexType>

<xs:sequence>

<xs:element maxOccurs="unbounded" name="Handle">

<xs:complexType>

<xs:sequence>

<xs:element name="HandleControlIndex" type="xs:unsignedByte" />

<xs:element name="HandleOperation" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Ở giai đoạn load, do chỉ lưu chỉ số của control nên việc dựng lại các sự kiện phải được thực hiện sau khi *việc load các control đã hoàn tất*. Việc load lại này khá đơn giản, ta chỉ việc duyệt từng thẻ tên là Event, lấy các thông tin của sự kiện đã lưu và gọi hàm *RegisterEvent* của đối tượng *MDTEventManager*.

## Các vấn đề riêng đối với Silverlight

### Hỗ trợ nội dung HTML trong Silverlight

* Vấn đề

Nội dung web dạng HTML hiện nay vô cùng phong phú, đặc biệt là gần đây, với trào lưu web 2.0, các ứng dụng widget dựa trên javascript và html xuất hiện ngày càng đa dạng. Do đó, xuất hiện nhu cầu hỗ trợ html trong Silverlight.

Không chỉ vậy, bản thân Silverlight không hiện thị được nội dung flash. Nếu có thể hỗ trợ html thì có thể dùng thẻ object trong HTML để hiển thị flash trong Silverlight.

* Giải pháp

Bản Silverlight 4 mới nhất đã có *WebBrowser* control hỗ trợ duyệt nội dung HTML. Tuy nhiên để sử dụng được control này, thì ứng dụng phải chạy với chế độ Out of Browser. Chế độ Out of Browser đòi hỏi người dùng phải cài đặt ứng dụng xuống Desktop, và chạy như một ứng dụng Desktop thông thường. Nếu muốn hiển thị nội dung HTML khi chạy trên trình duyệt web thì không được.

Nhóm chúng em tìm hiểu thì biết được có thể hiện thị nội dung HTML bằng cách chèn trực tiếp thẻ div hoặc iframe vào trong trang web chứa ứng dụng Silverlight. Tuy nhiên, khi chèn trực tiếp thẻ div hoặc iframe thì tọa độ sẽ tính theo tọa độ trang web, không còn đúng với hệ thống layout trong Silverlight. Do đó phải tính lại tọa độ và cập nhật lại cho thẻ div hoặc iframe.

* **Cách thực hiện:**
* Tạo thẻ div:

HtmlElement divIFrameHost = HtmlPage.Document.CreateElement("div");

* Chọn cách định vị của thẻ div là tọa độ tuyệt đối:

divIFrameHost.SetStyleAttribute("position", "absolute");

* Gán vị trí của thẻ div:

divIFrameHost.SetStyleAttribute("left", left + "px");

divIFrameHost.SetStyleAttribute("top", top + "px");

* Gán chiều dài, chiều rộng của thẻ div:

divIFrameHost.SetStyleAttribute("height", height + "px");

divIFrameHost.SetStyleAttribute("width", width + "px");

* Tạo iframe:

HtmlElement iframe = HtmlPage.Document.CreateElement("IFRAME");

* Thêm thẻ div vào trang web:

HtmlPage.Document.Body.AppendChild(divIFrameHost);

* Thêm thẻ div vào trang web:

HtmlPage.Document.Body.RemoveChild(divIFrameHost);

* Thêm iframe vào thẻ div:

divIFrameHost.AppendChild(iFrame);

* Gỡ bỏ iframe khỏi thẻ div:

divIFrameHost.RemoveChild(iFrame);

* Thay đổi src của iframe: dùng hàm Eval để thực thi đoạn mã javascript

StringBuilder script = new StringBuilder();

script.Append("var iframe = document.getElementById(\"" + str\_iFrame\_id + "\");");

script.Append("iframe.src = \"" + url + "\";");

HtmlPage.Window.Eval(script.ToString());

* Thay đổi nội dung của iframe: dùng hàm Eval để thực thi đoạn mã javascript

StringBuilder script = new StringBuilder();

script.Append("var doc = document.getElementById(\"" + str\_iFrame\_id + "\").contentDocument;");

script.Append("doc.open();");

script.Append("doc.write(\"<div>" + content.Replace("\"", "\\\"") + "</div>\");");

script.Append("doc.close();");

HtmlPage.Window.Eval(script.ToString());

### Truy xuất tên miền bên ngoài (cross-domain ])

* Vấn đề

Mặc định là Silverlight và các ứng dụng web client-side như Ajax, Flash… đều bị hạn chế là chỉ được truy xuất tên miền chứa ứng dụng, không được truy xuất tên miền bên ngoài. Ví dụ: ứng dụng đặt tại địa chỉ [*http://www.abc.com/silverlight.html*](http://www.abc.com/silverlight.html) thì không truy xuất được [*http://www.def.com/news.rss*](http://www.def.com/news.rss).

Nguyên nhân của việc hạn chế này là vì vấn đề bảo mật. Chẳng hạn, nếu không có sự hạn chế này, một ứng dụng web client-side có thể sử dụng client để tấn công từ chối dịch vụ (DoS) một server khác. Ví dụ: khi người dùng truy cập ứng dụng Silverlight [*http://www.abc.com/silverlight.html*](http://www.abc.com/silverlight.html)*,* ứng dụng này lại liên tục thực hiện hàng loạt truy xuất đến server [*http://www.def.com*](http://www.def.com)*.*Và có rất nhiều người dùng ứng dụng Silverlight này thì server đó sẽ không chịu nổi một lượng truy cấp lớn như vậy.

Do đó, ứng dụng muốn truy xuất server bên ngoài thì phải có sự cho phép của server đó, cụ thể là server chứa ứng dụng Silverlight phải nằm trong danh sách cho phép truy cập tại một trong hai file cấu hình sau của server cần lấy dữ liệu:

* *crossdomain.xml*: là file cấu hình của Flash nhưng được Silverlight hỗ trợ.
* *clientaccesspolicy.xml*: là file cấu hình của riêng Silverlight, hỗ trợ nhiều chức năng hơn file *crossdomain.xml*.

Tuy nhiên, không phải server nào cũng có file cấu hình như vậy và nếu có thì cũng có thể server đó không cho ứng dụng của ta truy xuất đến. Do đó, vấn đề đặt ra là làm sao có thể truy xuất được server bên ngoài.

* Giải pháp

Nhóm chúng em tìm hiểu thì biết được giải pháp chung cho vấn đề này là dùng service trung gian chạy trên server, vì server truy xuất server thì không gặp hạn chế cross-domain.

* Giải pháp 1

Dùng Web Services (service dạng SOAP). Với giải pháp này thì ta sẽ dựng một Web Services (WCF, WCF RIA …) để cung cấp các hàm truy xuất dữ liệu. Đồng thời, đặt file cấu hình tại server chứa Web Services để cho phép ứng dụng Silverlight truy xuất. Sau đó, ứng dụng Silverlight sẽ truy xuất các hàm của Web Services để lấy dữ liệu từ server bên ngoài.

* Ưu điểm
* Được Visual Studio hỗ trợ rất tốt, chỉ cần *Add web reference* là có thể sử dụng Web Services như một lớp bình thường.
* Khuyết điểm
* Gặp rắc rối khi truy xuất từ dll. Vì muốn truy xuất được Web Services thì cần một số lớp do Visual Studio phát sinh và các thông tin config. Thông tin config có hai cách để gán:
* Dùng file config: cách này sẽ gặp lỗi không tìm thấy file khi đem dll qua dùng tại project khác.
* Gán bằng code: Cách này chạy tốt nhưng thông tin về Web Services bị gán cứng. Nếu Web Services có gì thay đổi (đổi port…) thì dll không còn truy xuất được.
* Giải pháp 2

Dùng ASHX handler (service dạng REST). ASHX thực chất cũng tương tự trang aspx của ASP.NET, chỉ khác một chỗ là trang aspx trả về nội dung HTML để client hiển thị, còn trang ashx chủ yếu dùng để trả về dữ liệu cho client. Mỗi file ashx có công dụng như một hàm thực thi xử lý nào đó và trả về giá trị. Cách truyền tham số cho trang ashx là truyền theo dạng GET của HTML (?key=value&key=value…).

* Ưu điểm
* Mọi thứ trở nên rõ ràng, chỉ cần đường dẫn đến trang ashx là có thể dùng được service, không cần các thông tin hoặc lớp phụ nào khác.
* Cài đặt đơn giản.
* Khuyết điểm
* Phải tự nhận biết tham số cần truyền và cấu trúc giá trị trả về.

🗒 Nhận xét :

Giải pháp 1 rất tốt đối với ứng dụng thông thường, nhưng với trường hợp dùng cho dll thì gặp nhiều vấn đề. Do đó, nhóm chúng em chọn lựa giải pháp 2. Khuyết điểm của giải pháp 2 có thể giải quyết bằng cách bổ sung thêm lớp *Utility* cung cấp các hàm gọi các trang ashx và trả về giá trị cho người dùng. Khi sử dụng thì người dùng chỉ cần gọi hàm của lớp *Utility* như gọi hàm bình thường. Ngoài ra, lớp Utility có một biến *ServerURL* để lưu địa chỉ của server chứa ashx. Khi mang lớp Utility hoặc các file ashx qua server khác thì chỉ cần gán lại biến *ServerURL* để chỉ đúng server chứa ashx là sử dụng được service, không cần chỉnh sửa gì thêm.

* **Cách thực hiện:**
* Cấu trúc trang ashx: khi tạo trang ashx thì Visual Studio tự phát sinh sẵn lớp kế thừa từ *IHttpHandler*, trong đó gồm hàm *ProcessRequest* và thuộc tính *IsReusable*, ta chỉ cần chú ý hàm *ProcessRequest*. Đây chính là hàm thực hiện xử lý và trả kế quả về.

public void ProcessRequest(HttpContext context)

{

//thamso = context.Request["tenthamso"];

//thuc hien cac xu ly can thiet de lay du lieu.

//tra du lieu:

//dung cac ham write trong context.Response

//hoac ghi truc tiep vao context.Response.OutputStream

}

* Cách gọi trang ashx: dùng hàm *OpenReadAsync* của đối tượng *WebClient*

WebClient webClient = new WebClient();

webClient.OpenReadCompleted += new OpenReadCompletedEventHandler(webClient\_OpenReadCompleted);

thamso = "?key=value&key=value..."

Uri xmlUri = new Uri(\_ServerURL, "tentrang.ashx" + thamso);

webClient.OpenReadAsync(xmlUri);

# Tổng quan toàn bộ hệ thống

*✍ Nội dung chương này sẽ giới thiệu tính năng của toàn bộ hệ thống đồng thời mô tả kiến trúc của toàn hệ thống. Ứng với mỗi phân hệ chính sẽ có những mô tả tổng quát.*

## Giới thiệu tổng quan

Hệ thống được xây dựng trên nhiều phân hệ riêng lẻ nhằm tạo ra khả năng thay đổi và nâng cấp một cách nhanh chóng tại một phân hệ mà không ảnh hưởng đến phân hệ khác. Đồng thời, cũng cung cấp sự kiện tiện lợi cho người dùng. Ngoài các tính năng của một công cụ hỗ trợ thiết kế trực quan rất dễ sử dụng, hệ thống còn hỗ trợ thêm một số tính năng khác nhằm đáp ứng những nhu cầu của người dùng bình thường.

Sau đây là những tính năng chính của hệ thống:

* Giao diện trực quan sinh động dễ sử dụng
* Hỗ trợ thiết kế trực quan
* Cho phép chọn lựa và thay đổi hiệu ứng cho control cũng như điều chỉnh thông số cho hiệu ứng đó
* Cho phép gắng kết sự kiện giữa các control một cách trực quan trên giao diện
* Hỗ trợ việc liên kết dữ liệu từ một nguồn xml hoặc từ cơ sở dữ liệu với control ListData một cách trực quan.
* Có khả năng mở rộng số lượng control và hiệu ứng cho control.
* Cho phép xem trước ứng dụng đã thiết kế
* Trình diễn lại ứng dụng đã thiết kế tại domain của hệ thống
* Chỉnh sửa ứng dụng đã thiết kế

## Kiến trúc hệ thống

Toàn bộ hệ thống bao gồm bốn thành phần chính:

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **Design Core** | Đây là thành phần cung cấp công cụ hỗ trợ thiết kế trực quan ứng dụng Silverlight trên giao diện. Người dùng có thể tạo control, cho hiệu ứng, gắng kết sự kiện hoặc liên kết dữ liệu… dễ dàng trên giao diện của công cụ. Đồng thời cũng cung cấp chức năng xem trước ứng đã được thiết kế. |
| **Present Core** | Thành phần này được dùng để trình diễn lại ứng dụng Silverlight đã được thiết kế và lưu trữ tại cơ sở dữ liệu của hệ thống. |
| **Data Service** | Đây là thành phần dùng để cung cấp dữ liệu cho ***Design Core*** và ***Present Core*** thông qua các service server. Dữ liệu có thể là dữ liệu từ cơ sở dữ liệu của hệ thống hoặc từ nguồn khác. |
| **Component Downloader** | Đây là thành phần dùng để tại các component (dll) được lưu trữ trên server về client hỗ trợ cho việc thiết kế hoặc trình diễn lại. |

Bảng ‑ Các thành phần chính của hệ thống

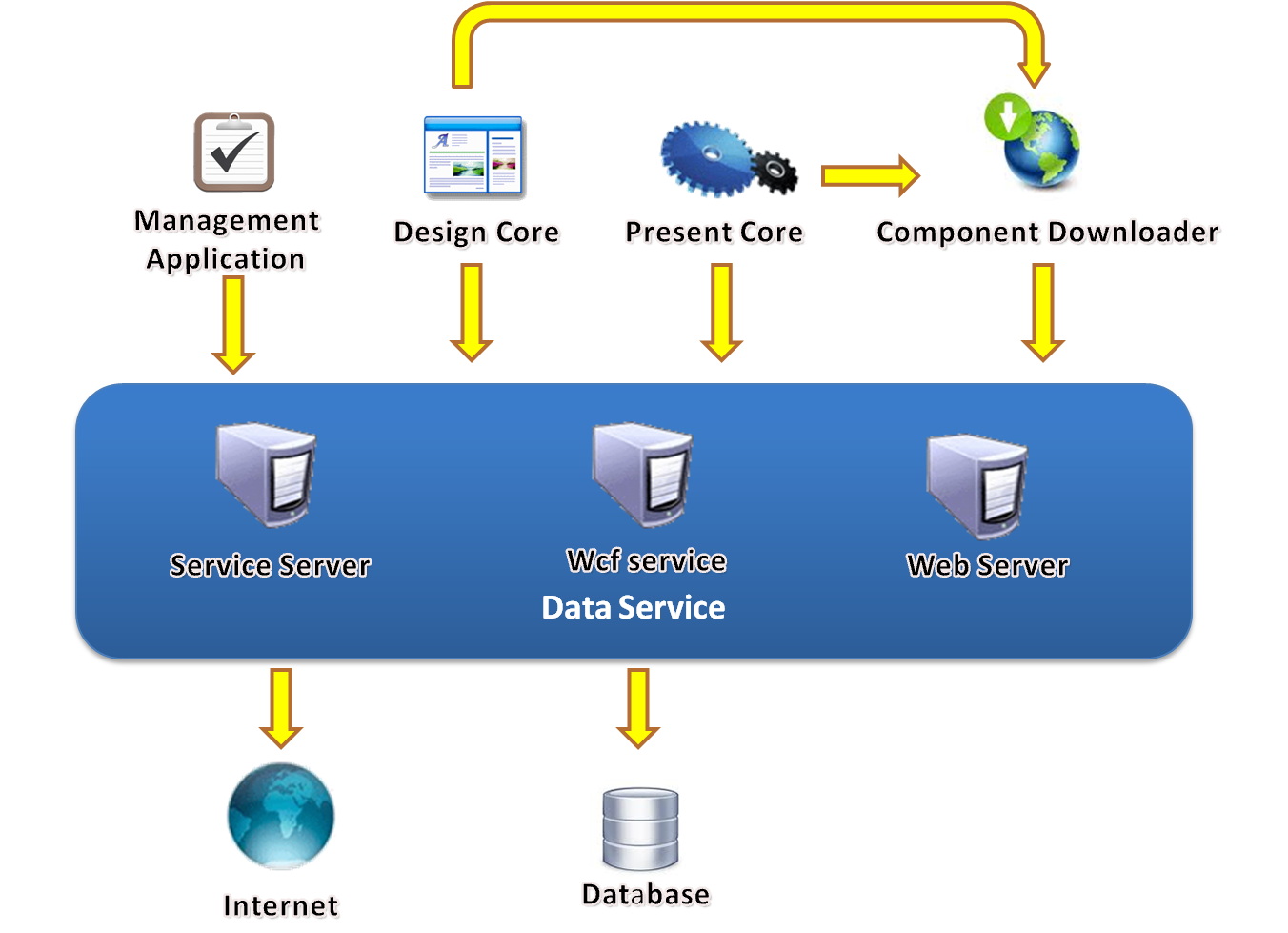
Các thành phần của hệ thống được xây dựng trên hai nền tảng là Silverlight 4 và asp.NET nhằm tạo ra tính tiện dụng, phối hợp hoạt động một cách tốt nhất giữa các tầng, cũng như là khắc phục những hạn chế của Silverlight. Toàn bộ hệ thống có thể được phân thành ba tầng bao gồm Data, Data Service, Application.

Sau đây là nhiệm vụ của từng tầng:

Tầng **Data**, đây là tầng dữ liệu của hệ thống, chứa cơ sở dữ liệu của toàn bộ hệ thống. Nơi đây sẽ lưu trữ về các thông tin định danh về người sử dụng, thông tin về các ứng dụng của người sử dụng đó.

Tầng Data Service, đây là nơi cung cấp dữ liệu cho tầng **Application**. Dữ liệu có thể từ cơ sở dữ liệu của hệ thống hoặc từ một nguồn nào đó (tập tin xml hoặc cơ sở dữ liệu) trên Internet, đồng thời nơi đây cũng là nơi cung cấp các tập tin dll (component) mà tầng **Application** yêu cầu. Ngoài ra, tầng này cũng có nhiệm vụ ghi thông tin vào cơ sở dữ liệu của hệ thống khi được yêu cầu.

Tầng **Application**, đây là tầng cao nhất của hệ thống, được dùng để tương tác với người dùng. Công cụ thiết kế (***Design Tool***) và trình diễn (***Present***) là hai ứng dụng chính của tầng này. Bên cạnh đó, còn một ứng dụng nhỏ để người dùng quản lý các ứng dụng mà họ đã thiết kế.



Hình ‑ Mô hình kiến trúc hệ thống

### Tầng Data

Hệ thống sử dụng hệ quản trị cơ sở dữ liệu Microsoft SQL Server 2005 để lưu trữ và quản lý dữ liệu của hệ thống.



Hình ‑ Nội dung lưu trữ tầng Data

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **User information** | Lưu trữ các thông tin về người dùng như tên đăng nhập, mật khẩu, email… |
| **Management Information** | Lưu trữ các thông tin về ứng dụng mà người dùng đã thiết kế |
| **Designed Application Xml Data** | Đây là chuỗi xml để mô tả về một ứng dụng đã được thiết kế. Chuỗi xml này chứa thông tin về các dll cần dùng, các control và hiệu ứng sẽ được tạo ra và các thông tin về gắng kết sự kiện giữa các control |

Bảng ‑ Dữ liệu ở tầng Data



Hình ‑ Lược đồ cơ sở dữ liệu ở tầng Data

### Tầng Data Service

Đây là nơi cung cấp dữ liệu cho các ứng dụng ở tầng **Appication** và đảm nhận nhiệm vụ ghi nhận vào cơ sở dữ liệu khi được yêu cầu. Dữ liệu có thể được đọc lên từ cơ sở dữ liệu hoặc từ trên Internet. Nói cách khác, tầng này có nhiệm vụ như một trạm trung chuyển (proxy) dữ liệu từ cơ sở dữ liệu hoặc Internet cho các ứng dụng. Mục đích của tầng này là nhằm giải quyết vấn đề *cross-domain*.



Hình ‑ Các thành phần ở tầng Data Service

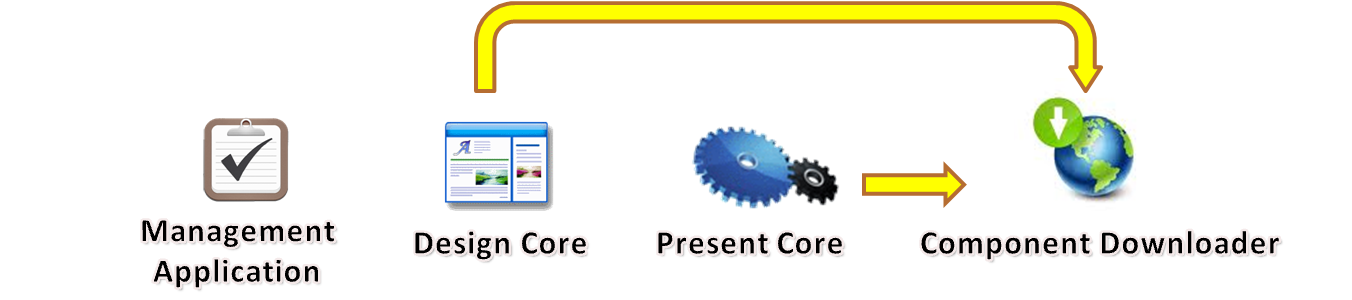
|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **Service Server** | Máy chủ sẽ cung cấp các dịch vụ nhằm hỗ trợ việc truy xuất dữ liệu từ bên ngoài Internet. Nguồn dữ liệu có thể là một tập tin xml hoặc là từ cơ sở dữ liệu của server khác. Khi dữ liệu được đưa qua máy chủ này sẽ được định dạng lại thành chuỗi XML hoặc List<List<string>> mới được về cho ứng dụng. |
| **Wcf Service** | Cung cấp dữ liệu được đọc lên từ cơ sở dữ liệu và ghi nhận dữ liệu vào cơ sở dữ liệu của hệ thống. |
| **Web Server** | Cung cấp các tập tin dll cho ứng dụng. |

Bảng ‑ Các thành phần ở tầng Data Service

* **Service Server:** Máy chủ để cung cấp các dịch vụ truy xuất dữ liệu từ bên ngoài. Máy chủ này có nhiệm vụ giải quyết vấn đề cross-domain trong ứng dụng đồng thời cũng là nơi trung gian để chuyển đổi dữ liệu về dạng mà hệ thống hiểu được. Thành phần này được phát triển trên nền tảng asp.NET. Mỗi dịch vụ của thành phần này cung cấp sẽ tương ứng với một địa chỉ url đến tập tin .ashx trên server. Cách sử dụng tập tin aspx được mô tả ở mục 4.3.2.
* **Wcf Service:** là dịch vụ cung cấp các hàm để lấy các thông tin liên quan đến cơ sở dữ liệu như lấy danh sách các ứng dụng đã được thiết kế bởi người dùng (*GetDesignedApplicationList*), lấy thông tin về một ứng dụng có trong cơ sở dữ liệu (*GetDesignedApplication)*… thành phần này được phát triển dựa trên nền tảng **Windows Communication Foundation** (WCF) kết hợp với việc sử dụng **Linq To Sql** để truy xuất nhanh chóng cơ sở dữ liệu của hệ thống.
* **Web Server:** thành phần công cấp các component cho client. Mỗi component sẽ tương ứng với một địa chỉ url trên server. Các component này là các tập dll được lưu trữ trên server với cấu trúc được mô tả ở mục 4.1.2.

### Tầng Application

Nơi đây sẽ cung cấp các ứng dụng mà hệ thống hỗ trợ cho người dùng bao gồm: công cụ hỗ trợ thiết kế, ứng dụng trình diễn, và một ứng dụng để quản lý các ứng dụng đã thiết kế cho người dùng.



Hình ‑ Các thành phần ở tầng Application

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **Design Core** | Công cụ hỗ trợ thiết kế trực quan cho người dùng. Với công cụ này người dùng sẽ dễ dàng và nhanh chóng tạo ra một ứng dụng Silverlight chỉ bằng các thao tác đơn giản trên giao diện thiết kế. Ứng dụng còn cho phép xem trước ứng dụng đã thiết kế. |
| **Present Core** | Ứng dụng dùng để trình diễn lại ứng dụng đã được người dùng thiết kế. |
| **Management Application** | Đây là ứng dụng giúp cho người dùng có thể quản lý được các ứng dụng họ đã thiết kế và đang được lưu trữ trên server. |
| **Component Downloader** | Đây là thành phần dùng để tải các dll cần dùng về client. |

Bảng ‑ Các thành phần ở tầng Application

Tất cả các thành phần đều được phát triển trên nền công nghệ Silverlight 4 nhằm đảm bảo tính tương thích giữa các thành phần.

* **Design Core:** Đây là thành phần chính của phân hệ thiết kế. Thành phần này một công cụ có khả năng hỗ trợ thiết kế trực quan ứng dụng Silverlight. Thành phần này có nhiệm vụ hỗ trợ cho người dùng xây dựng nhanh một ứng dụng Silerlight, tạo các control và hiệu ứng trên control đó, điều chỉnh các thuộc tính của control và hiệu ứng đã tạo, gắng kết xử lý giữa các control… Ngoài ra, thành phần này còn có nhiệm vụ load lại ứng dụng đã thiết kế được lưu trữ trên cơ sở dữ liệu của hệ thống, đồng thời gửi yêu cầu cho tầng DataService để lưu trữ lại ứng dụng người dùng đang thiết kế.
* **Present Core:** Đây là thành phần được dùng để trình diễn lại ứng dụng mà người dùng yêu cầu. Thành phần có nhiệm vụ dựng lại ứng dụng và quản lý quá trình hoạt động của ứng dụng đó.
* **Component Download:** là thành phần dùng để tải các component (dll) lúc runtime về client nhằm hỗ trợ cho **Design Core** và **Present Core**. Đồng thời, thành phần này cũng sẽ quản lý các component đã được tải về.
* **Management Application:** là ứng dụng được cung cấp để hỗ trợ cho người dùng quản lý các ứng dụng mà họ đã thiết kế. Ứng dụng này phát triển dựa trên template **Business Application** của Visual Studio. Với ứng dụng này, người dùng có thể xem, chỉnh sửa hoặc xóa đi những ứng dụng mà họ đã thiết kế.

## Kết luận

Trong chương này chúng em đã trình bày tổng quan về kiến trúc toàn bộ hệ thống và các thành phần trong từng tầng của hệ thống. Với kiến trúc đã trình bày cho thấy khả năng nâng cấp và chỉnh sửa các thành phần trong hệ thống mà không ảnh hưởng đến các thành phần khác. Với khả năng này, hệ thống sẽ dễ dàng được mở rộng và cải tiến về sau này.

# Phân hệ thiết kế

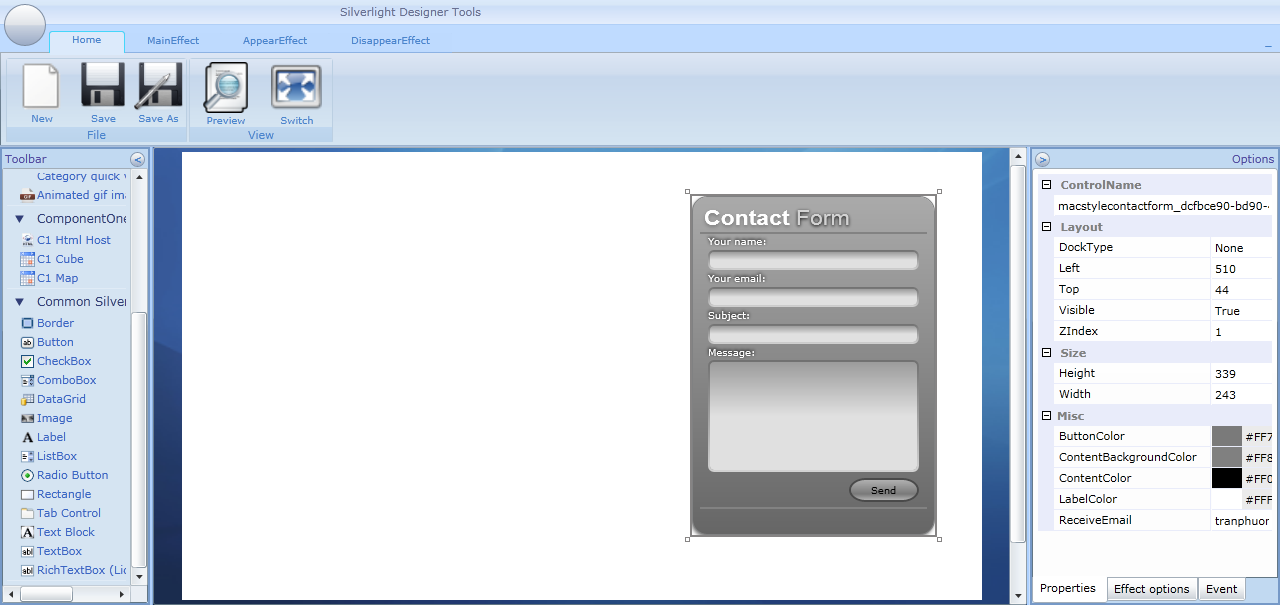
*✍ Nội dung của chương này trình bày chi tiết qui trình thực hiện, kiến trúc hệ thống và các tính năng của phân hệ thiết kế cộng với các dịch vụ mà hệ thống này cung cấp để hỗ trợ người dùng thiết kế ứng dụng.*

## Giới thiệu

Đây là phân hệ cung cấp công cụ hỗ trợ người dùng thiết kế trực quan ứng dụng Silverlight. Với công cụ này, người dùng sẽ dễ dàng và nhanh chóng tạo ra ứng dụng Silverlight mong muốn bằng những thành phần mà hệ thống hỗ trợ.

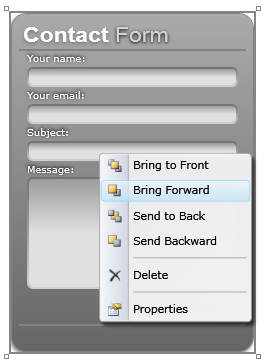
Cách thức bố cục giao diện tương tự như giao diện của Visual Studio quen thuộc giúp tạo sự thân thiện và dễ dùng cho người sử dụng. Phân hệ gồm như thành phần chính sau:

* **DesignCanvas:** đây là nơi để thiết kế ứng dụng Silverlight.
* **PropertyGrid:** nằm bên phải Design Canvas, cung cấp giao diện cho phép thay đổi các thuộc tính của control đang chọn.
* **ControlTree:** nằm bên trái Design Canvas, cung cấp danh sách các control mà hệ thống hỗ trợ đang có để hỗ trợ người dùng thiết kế ứng dụng.
* **RibbonMenu:** nằm bên trên Design Canvas, cung cấp danh sách các hiệu ứng cho người dùng chọn lựa, các chức năng save/load ứng dụng, đồng thời cũng cung cấp các chức năng để gọi editor tương ứng loại control. Với editor được cung cấp thì người dùng thao tác dễ dàng hơn với control.

****

Hình ‑ Giao diện chính của công cụ

Ngoài ra, khi người dùng nhấn chuột phải vào một control nào đó thì sẽ xuất hiện một trình đơn ngữ cảnh (context menu).



Hình ‑ Trình đơn ngữ cảnh (context menu)

Với trình đơn ngữ cảnh này, người dùng có thể sử dụng các chức năng Arrange cho control bao gồm *Bring To Font*, *Bring Forward, Send To Back, Send Backward*… Đồng thời, người dùng cũng có thể chọn *Delete* từ trình đơn này để xóa bỏ control hoặc chọn *Properties* để gọi hiện PropertyGrid nếu như PropertyGrid đang bị ẩn đi.

Bên cạnh trình đơn ngữ cảnh, các editor cũng là một phần không thể thiếu của công cụ. Các editor này sẽ cung cấp một giao diện tiện dụng cho người dùng thao tác với control một cách nhanh chóng.

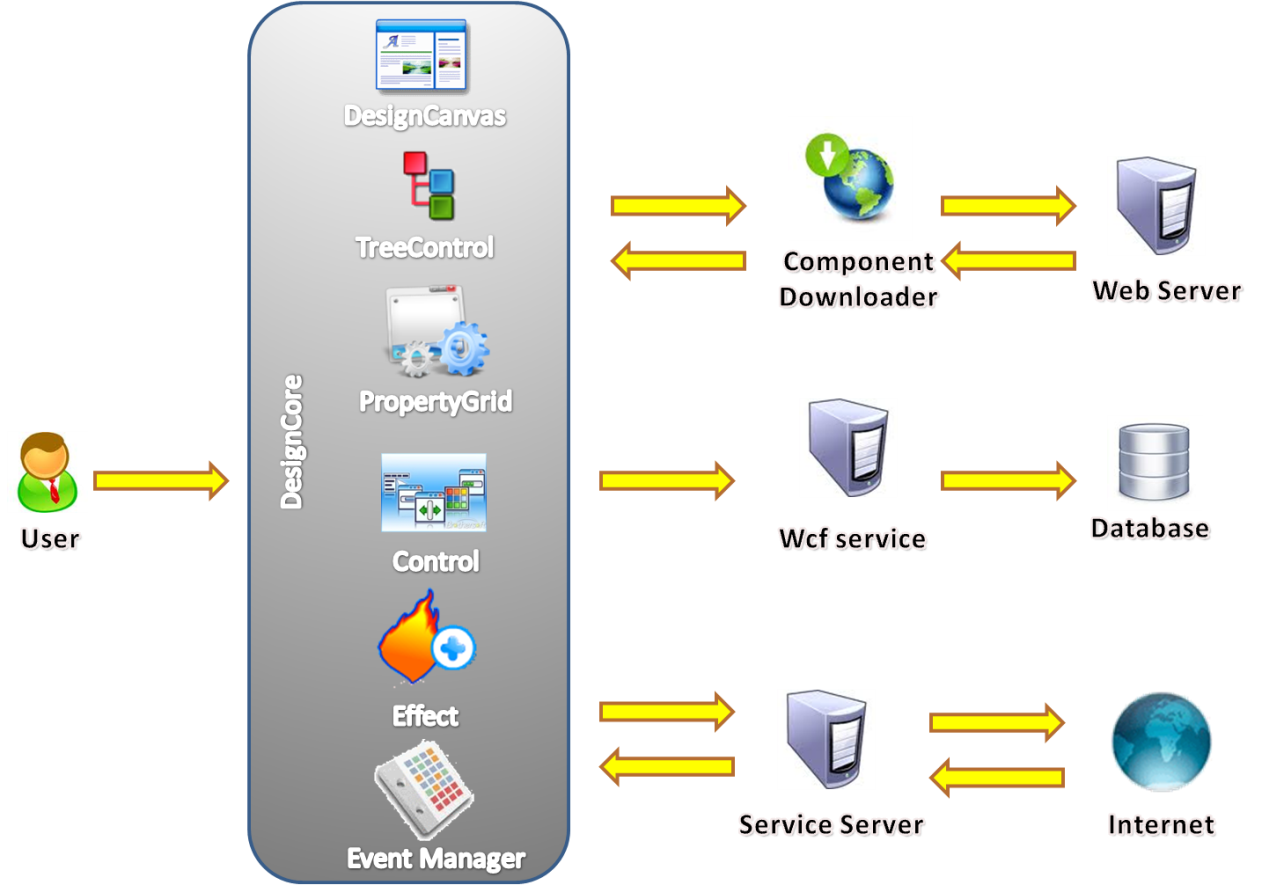
Một đặc điểm không thể không nhắc đến của phân hệ này đó là khả năng mở rộng số lượng control và hiệu ứng của công cụ. Với khả năng này, lượng control và hiệu ứng mà công cụ hỗ trợ sẽ không có giới hạn, bởi vì có thể bổ sung thêm bất cứ lúc nào những control và hiệu ứng mới sau khi đã triển khai công cụ.

## Qui trình thiết kế một ứng dụng

1. Tạo ra một ứng dụng từ Menu.
2. Chọn lựa control cần dùng từ ControlTree, sau đó double click vào control cần tạo để yêu cầu công cụ tạo control đó.
3. Sử dụng các thao tác drag drop để điều chỉnh vị trí hoặc kích thước của control. Đồng thời sử dụng PropertyGrid và các Editor (nếu có) đi kèm với control để điều chỉnh các thuộc tính khác của control.
4. Chọn lựa hiệu ứng cho control và hiệu chỉnh thuộc tính của hiệu ứng. Mặc định, mỗi control đều có *MainEffect* (hiệu ứng chính của control), *AppearEffect* (hiệu ứng khi control xuất hiện) và *DisappearEffect* (hiệu ứng khi control mất đi). Hơn nữa, nếu là control list thì có thêm *ListEffect* và *ListItemEffect*. Ngoài ra, có thể có thêm một số hiệu ứng tùy thuộc vào khả năng của control hỗ trợ.
5. Chọn datasource, kiểu item và liên kết dữ liệu cho control nếu control là DataList.
6. Gắng kết sự kiện control phát sinh với hàm xử lý để phối hợp hoạt động của các control.
7. Lặp lại bước 2 🡪 bước 6 cho đến khi hoàn thành ứng dụng Silverlight mong muốn.
8. Dùng chức năng preview để xem lại ứng dụng đã thiết kế.
9. Save ứng dụng xuống cơ sở dữ liệu để trình diễn lại hoặc để lần sau có thể chỉnh sửa.

## Kiến trúc

Phân hệ được phát triển sử dụng công nghệ Silverlight 4 phiên bản mới nhất nhằm tận dụng tốt nhất những tính năng của Silverlight. Đồng thời nhằm tạo ra một công cụ hỗ trợ thiết kế một cách trực quan, cũng như giúp cho công cụ có thể dể dàng được sử dụng mọi nơi. Sau đây là mô hình và mô tả chi tiết các thành phần trong phân hệ này.



Hình ‑ Kiến trúc phân hệ thiết kế

Nhiệm vụ chính của phân hệ này là cung cấp một công cụ hỗ trợ thiết kế trực quan ứng dụng Silverlight.

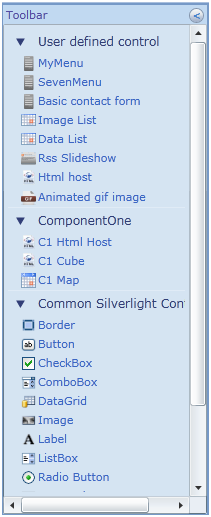
Người dùng sẽ thực hiện các thao tác tạo control, drag drop, hiệu chỉnh kích thước, hiệu ứng... của control với **DesignCore**. Khi cần dữ liệu bên ngoài như file rss, dữ liệu cho control list… sẽ gọi **ServiceServer.** Ngoài ra, để tăng tốc độ load ứng dụng (giảm dung lượng file cần download) đồng thời cũng để giúp cho hệ thống có khả năng mở rộng số lượng control và hiệu ứng về sau, chương trình được viết theo dạng component (các control và hiệu ứng được biển dịch (build) thành dll), ban đầu chỉ load thông tin để hiển thị, khi nào cần dùng thì mới tiến hành download thông qua **ComponentDownloader**.

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **DesignCore** | Thành phần chính của hệ thống, cung cấp tất cả các tính năng phục vụ cho việc thiết kế. |
| **ServiceServer** | Cung cấp dịch vụ truy xuất dữ liệu bên ngoài để khắc phục vấn đề Cross-domain |
| **Component Downloader** | Phục vụ cho việc download các thành phần cần dùng cho chương trình. |
| **Controls** | Các control được tạo ra trong ứng dụng, dùng để tương tác với người sử dụng. |
| **Effects** | Các hiệu ứng được gán cho các control được tạo nhằm tạo ra một giao diện sinh động, đẹp mắt |
| **Event Manager** | Đây là thành phần theo dõi các sự kiện được phát sinh từ các control. Ngoài ra, thành phần này có còn nhiệm vụ phân phối sự kiện cho các control. |

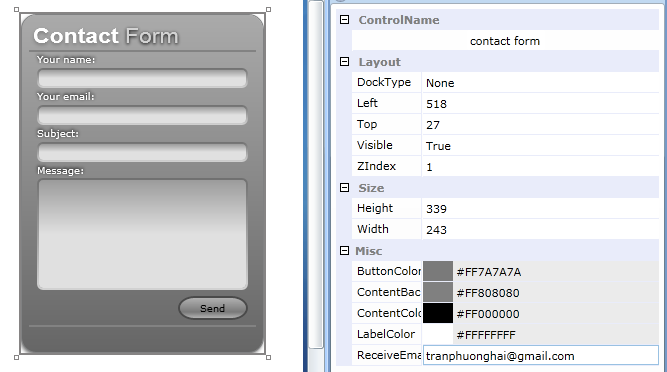
Bảng ‑ Các thành phần của phân hệ thiết kế

## Quy trình sử dụng

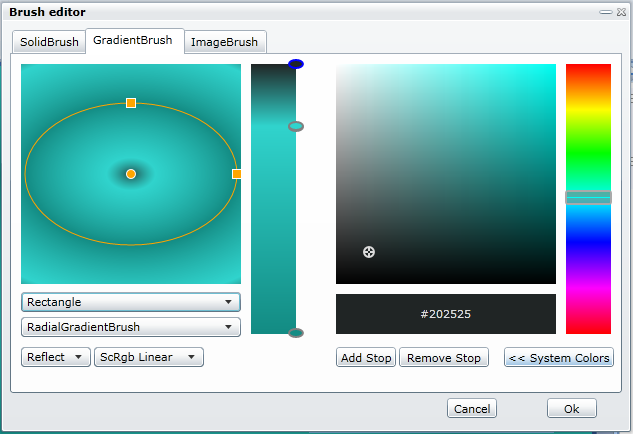
1. Chọn chức tạo mới ứng dụng Silverlight bằng cách chọn **New** trên **Menu Ribbon.**
2. Từ ControlTree, lựa chọn những control muốn sử dụng. Sau khi đã chọn được control ưng ý thì double click vào control đó để yêu cầu hệ thống công cụ tạo ra control. Khi control được tạo ra lần đầu tiên thì cần phải đợi công cụ tải những thành phần cần thiết kế để tạo ra control về, sau đó control mới được tạo ra. Ngược lại thì control sẽ được tạo ra ngay lập tức.
3. Tùy chỉnh các thuộc tính của control bằng cách các thao tác drag drop, thay đổi trong PropertyGrid.



Hình ‑ Giao diện ControlTree thể hiện danh sách control được công cụ cung cấp

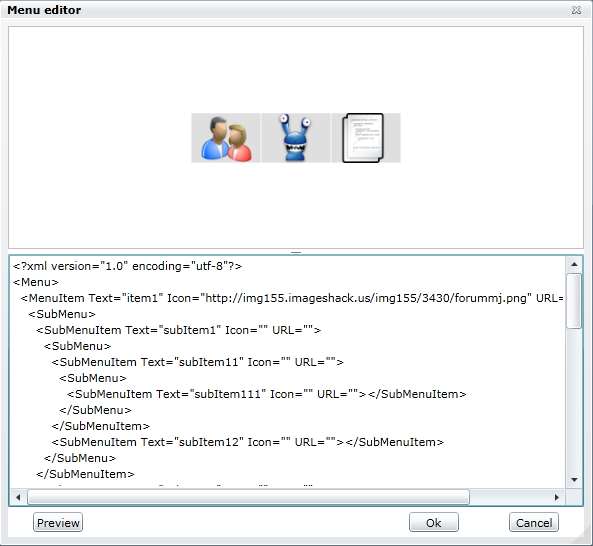


Hình ‑ Điều chỉnh thuộc tính của control qua Property Grid

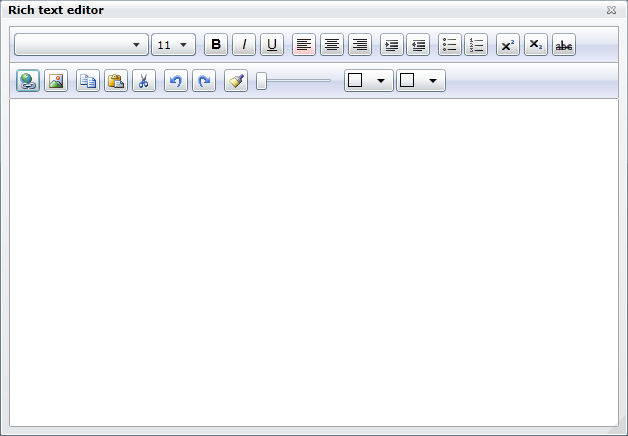


Hình ‑ Chọn lựa màu cho control thông qua BrushEditor

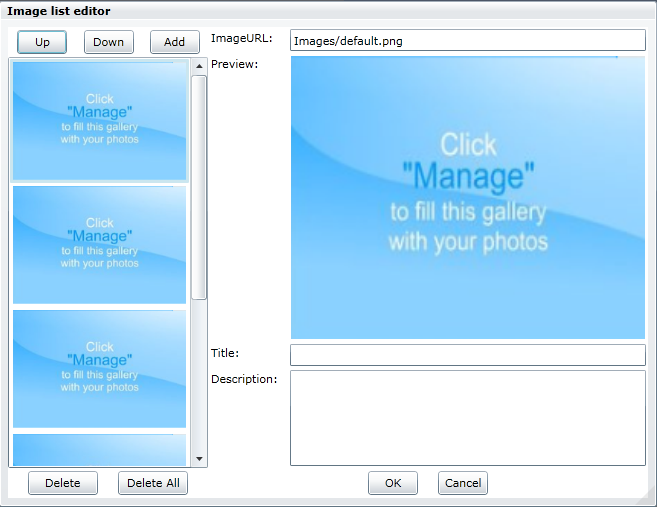
1. Sử dụng Editor nếu control có Editor đi kèm để gán dữ liệu, điều chỉnh các thuộc tính của control. Các hình Hình 6‑7, Hình 6‑8, Hình 6‑9 là giao diện của các editor đang được công cụ cung cấp.



Hình ‑ Cài đặt menu item bằng MenuEditor

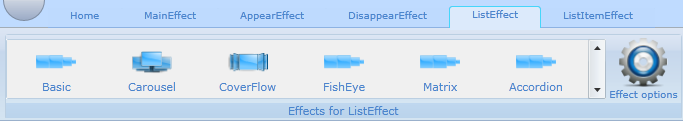


Hình ‑ Sử dụng RichTextEditor để định dạng dữ liệu để hiển thị cho RichText control



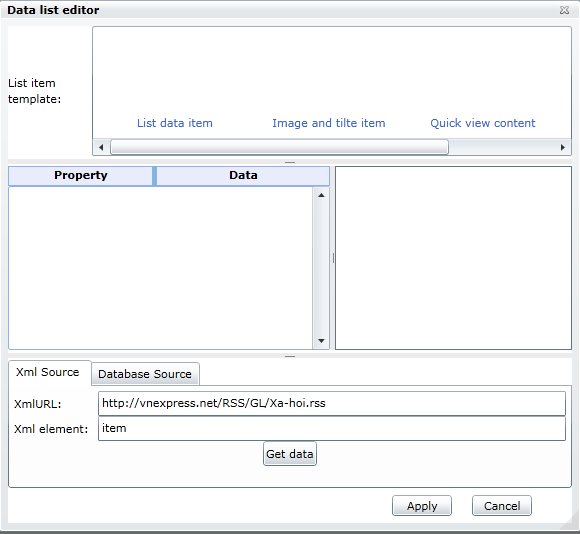
Hình ‑ Gán dữ liệu cho ImageList control thông qua ImageListEditor

1. Chọn hiệu ứng cho control và hiệu chỉnh thuộc tính của hiệu ứng được chọn. Khi muốn thay đổi hiệu ứng thì chỉ cần chọn vào biểu tượng hiệu ứng muốn thay đổi. Muốn tùy chỉnh tham số cho hiệu ứng thì chọn nút Effect Options ở bên phải. Sau khi chọn Effect Options thì PropertyGrid sẽ chuyển sang thẻ Effect. Tại thẻ này, danh sách thuộc tính có thể được hiệu chỉnh của một hiệu ứng sẽ được hiển thị trong một PropertyGrid.



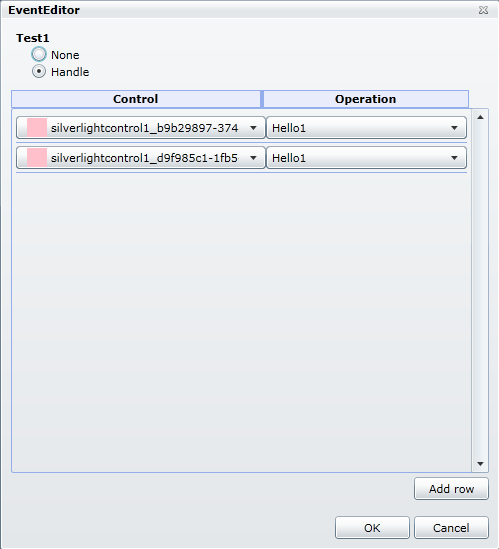
Hình ‑ Giao diện danh sách các hiệu ứng được cung cấp cho một loại hiệu ứng

1. Tiến hành liên kết (binding) dữ liệu cho control nếu control đó là DataList bằng cách dùng DataListEditor qua các bước sau:
   * Đầu tiên là chọn một ListItem template cần dùng. Danh sách các property của ListItem sẽ hiển thị tại cột Property.
   * Sau đó chọn nguồn của Datasource từ một tập tin xml hay là từ cơ sở dữ liệu. Nhấn Get data để công cụ tải dữ liệu về.
   * Khi data được nhận về sẽ hiển thị tại cột Data theo kiểu Combobox. Tiếp theo ta sẽ binding dữ liệu với item bằng cách chọn data cho property. Nếu property nào không dùng thì để \*\**\*Not use this property\*\*\**. Nhiều property có thể dùng chung một data.
   * Cuối cùng nhấn Apply.



Hình ‑ Giao diện của DataListEditor

1. Lặp lại bước 2 đến bước 5 cho đến khi đã tạo ra tất cả control cần dùng cho ứng dụng.
2. Tiến hành gắng kết sự kiện nhằm phối hợp xử lý giữa các control. Bằng cách sử dung EventEditor để chọn lựa các control và hàm xử lý tương ứng sẽ được cho một sự kiện, sau đó nhấn OK để đăng ký. Nếu như muốn gỡ bỏ việc gắng kết của một sự kiện nào đó thì trong EventEditor chọn radio button None và nhấn OK.



Hình ‑ Giao diện EventEditor

1. Sử dụng chức năng Preview để xem lại ứng dụng trước khi yêu cầu công cụ lưu trữ lại ứng dụng vừa thiết kế.

## Quy trình mở rộng công cụ

### Mở rộng control

Để thêm một control mới cho hệ thống, cần thực hiện nữa thao tác sau:

1. Add reference đến *BasicLibrary.dll.*
2. Nếu cài đặt một control hoàn toàn mới thì cần phải kế thừa lại từ control *BasicControl* hoặc là những control được kế thừa lớp cơ sở *BasicControl*. Nếu muốn thì một control mới vào hệ thống mà control đó không được kế thừa từ *BasicControl* thì phải bao bọc lại control đó trong một control kế thừa *BasicControl*.
3. Tạo ra các property cho control mới tạo và nếu muốn cho người dùng chỉnh sửa một property nào thì thêm tên của property đó vào trong biến *parameterNameList*.
4. Tạo ra các sự kiện control có thể phát sinh. Lưu ý các sự kiện này phải sử dụng delegate *MDTEventHandle*. Sau đó phải đăng ký sự kiện đó với hệ thống để hệ thống có thể quản lý sự kiện này bằng cách gọi hàm *AddEventNameToList* với tham số là tên sự kiện muốn đăng ký.
5. Tạo ra các hàm xử lý (operation) control cung cấp. Các operation chỉ nhận vào một tham số là chuỗi xml. Trong operation này cần phân tích xml đó để lấy dữ liệu cần thiết. Sau đó đăng ký với hệ thống bằng hàm *AddOperationNameToList* với tham số tên hàm xử lý muốn đăng ký.
6. Tạo ra các con trỏ hiệu ứng nếu cần cho control. Mặc định thì BasicControl đã hỗ trợ sẵn ba loại con trỏ hiệu ứng là *MainEffect*, *AppearEffect*, *DisappearEffect*. Nếu muốn thêm một loại hiệu ứng thì tạo ra con trỏ hiệu ứng (property) rồi thêm tên của con trỏ hiệu ứng này vào biến *effectPropertyNameList*. Sau đó override lại hàm *ChangeEffect* để cài đặt việc khởi tạo hiệu ứng khi người dùng thay đổi hiệu ứng cho con tro hiệu ứng này.
7. Đóng gói control thành tập tin dll và chép dll này vào thư mục *ClientBin/Controls/ControlDll* trên WebServer, chép những dll mà control có tham chiếu đến vào thư mục *ClientBin/Controls/ReferenceDll*. Sau đó thêm thông tin của control vào tập tin *info.xml* trong thư mục *ClientBin/Controls*.

**Ví dụ:** mã nguồn cài đặt *SimpleButton* bằng cách bao bọc lại lại *Button* control mà Silverlight hỗ trợ:

*Tập tin SimpleButton.xaml:*

<bl:BasicControl x:Class="MyControl.SimpleButton"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:bl="clr-namespace:BasicLibrary;assembly=BasicLibrary"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

mc:Ignorable="d"

d:DesignHeight="300" d:DesignWidth="400">

<Grid x:Name="LayoutRoot" Background="White">

<Button Content="Button" HorizontalAlignment="Stretch" Name="button" VerticalAlignment="Stretch" Click="button\_Click" />

</Grid>

</bl:BasicControl>

*Tập tin SimpleButton.xaml.cs:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Shapes;

using BasicLibrary;

namespace MyControl

{

public partial class SimpleButton : BasicControl

{

public SimpleButton()

{

InitializeComponent();

// dang ky property

parameterNameList.Add("ButtonContent");

//dang ky su kien

AddEventNameToList("ButtonClicked");

}

// khai bao su kien

public MDTEventHandler ButtonClicked;

// khai bao property

public string ButtonContent

{

get { return button.Content.ToString(); }

set { button.Content = value; }

}

private void button\_Click(object sender, RoutedEventArgs e)

{

if (ButtonClicked != null)

{

// phat sinh su kien

string xml = "<Root><Click>true</Click><ControlName>" + this.controlName + "</ControlName></Root>";

ButtonClicked(this, xml);

}

}

}

}

*Nội dung thêm vào tập tin info.xml:*

<Control>

<Name>MyControl.SimpleButton</Name>

<DllFilename>SimpleButton.dll</DllFilename>

<DllReferences>

</DllReferences>

<DisplayName>Simple button</DisplayName>

<Description>Simple button control</Description>

<IconName>SimpleButton.png</IconName>

<Group>Button</Group>

</Control>

### Mở rộng hiệu ứng

Để thêm một hiệu ứng mới cho hệ thống, cần thực hiện nữa thao tác sau:

1. Add reference đến *BasicLibrary.dll.*
2. Nếu cài đặt hiệu ứng cho control đơn thì kế thừa từ *BasicEffect*. Nếu cài đặt hiệu ứng cho control list thì kế thừa từ *BasicListEffect*.
3. Cài đặt các thao tác cần thiết để thực hiện hiệu ứng. Nếu hiệu ứng cần thêm một số vật liệu (ví dụ: các rectangle phủ lên bề mặt của control), thì thêm vào trong *CanvasRoot* của control. CanvasRoot này là một Canvas chứa trong đó là control được tác động hiệu ứng.
4. Tạo ra các property cho hiệu ứng mới tạo và nếu muốn cho người dùng chỉnh sửa một property nào thì thêm tên của property đó vào trong biến *parameterNameList*.
5. Override lại các hàm sau đây *Start* (hàm gọi bắt đầu thực hiện hiệu ứng), *Stop* (hàm gọi ngừng thực hiện hiệu ứng), *DetachEffect* (hàm gỡ bỏ các vật liệu mà hiệu ứng đã thêm vào trong *CanvasRoot*), *SetSelfHandle*.
6. Đóng gói hiệu ứng thành tập tin dll và chép dll này vào thư mục *ClientBin/Effects/EffectDll* trên WebServer, chép những dll mà control có tham chiếu đến vào thư mục *ClientBin/Effects/ReferenceDll*. Sau đó thêm thông tin của control vào tập tin *info.xml* trong thư mục *ClientBin/Effects*.

**Ví dụ:** mã nguồn cài đặt hiệu ứng *WipeIn* kế thừa từ *BasicEffect* xin xem trong phụ lục A.

## Kết luận

Trong chương này đã giới thiệu về qui trình, kiến trúc và các chức năng của phân hệ thiết kế. Người dùng sử dụng phân hệ này để thiết kế nhanh ứng dụng Silverlight. Với đặc điểm là một ứng dụng web được phát triển bằng công nghệ Silverlight nên dễ dàng triển khai rộng rãi công cụ. Ngoài ra, chương này còn trình bày cách để có thể mở rộng thêm số lượng control và hiệu ứng cho công cụ.

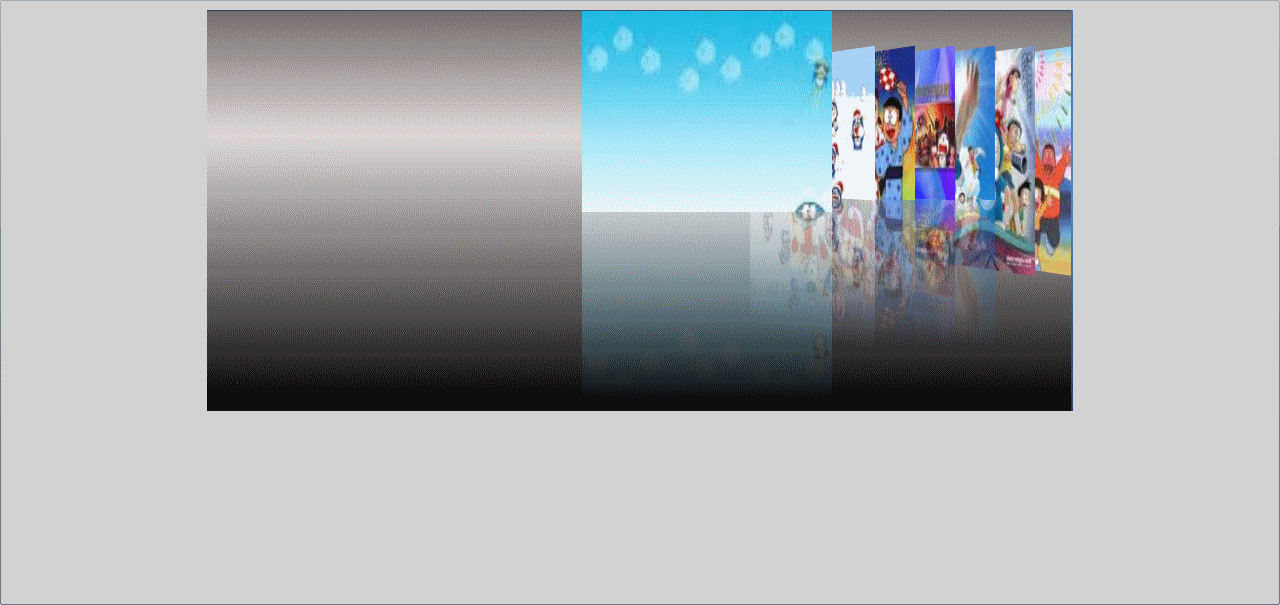
Ở chương sau sẽ trình bày phân hệ trình diễn, dùng để trình diễn lại ứng dụng đã được thiết kế.

# Phân hệ trình diễn

*✍ Nội dung của chương này trình bày chi tiết qui trình thực hiện, kiến trúc hệ thống và các tính năng của phân hệ trình diễn cộng với dịch vụ mà hệ thống này cung cấp để trình diễn lại ứng dụng đã thiết kế.*

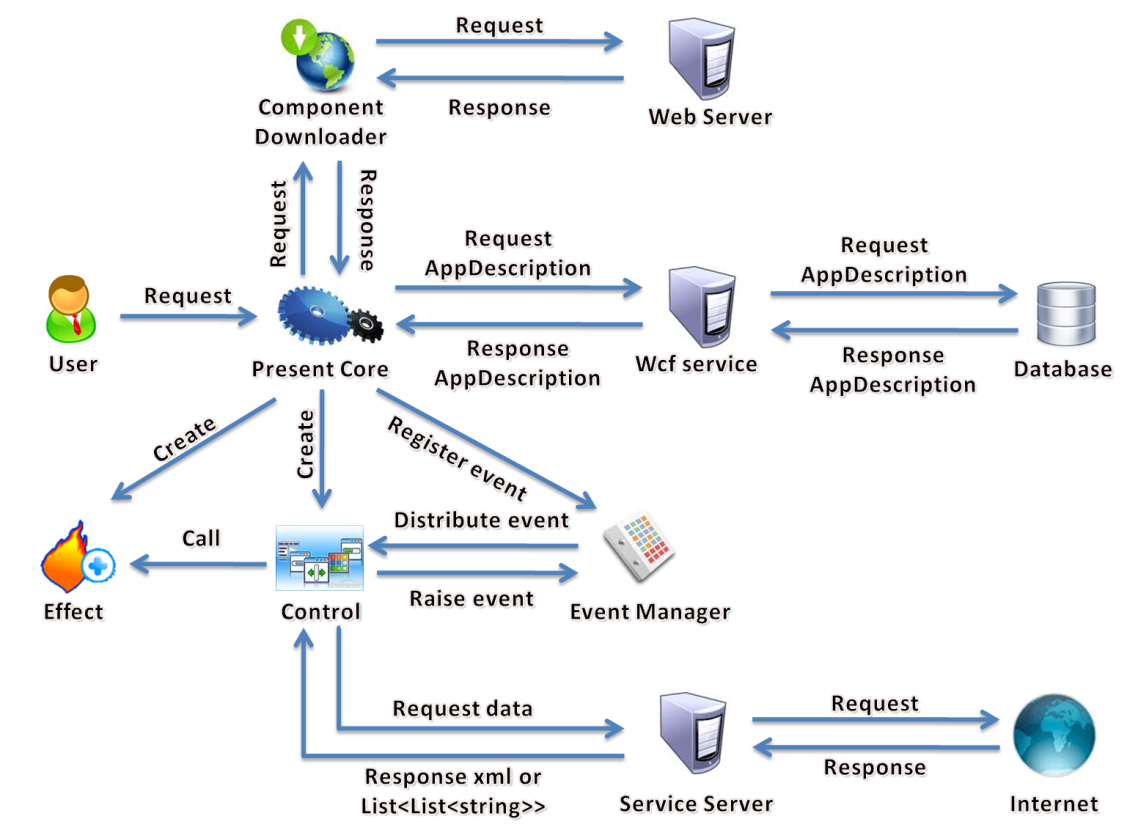
## Giới thiệu

Phân hệ này có tên là Present. Sau khi hoàn thành việc thiết kế ứng dụng và lưu trữ lại, người dùng đã có thể sử dụng tính năng của hệ thống. Phân hệ này được cài đặt trên nền công nghệ Silverlight 4 của Microsoft để có thể trình diễn lại ứng dụng Silverlight được yêu cầu. Phân hệ này khi hoạt động sẽ lấy nội dụng của ứng dụng từ cơ sở dữ liệu và tiến hành dựng lại và quản lý ứng dụng mà người sử dụng yêu cầu.



Hình ‑ Giao diện chính của ứng dụng Present

## Qui trình tạo lại và quản lý ứng dụng



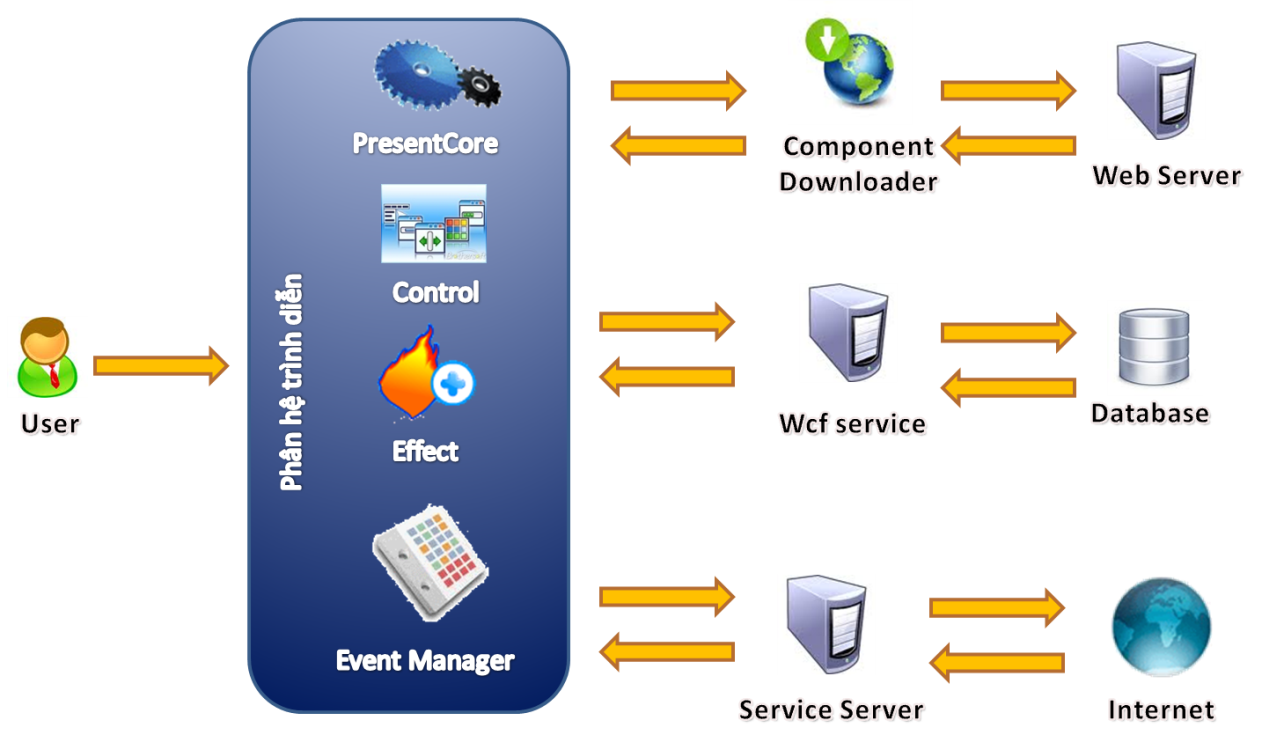
Hình ‑ Qui trình tạo lại và quản lý ứng dụng

Khi có yêu cầu tạo lại ứng dụng từ phía người dùng tới hệ thống. Hệ thống sẽ thực hiện các bước sau đây:

1. Tiến hành xác định định danh của ứng dụng được yêu cầu thông qua địa chỉ url.
2. Dựa vào định danh của ứng dụng, hệ thống sẽ tiến hành tải chuỗi XML mô tả về ứng dụng được lưu cơ sở liệu.
3. Sau khi tải về, hệ thống tiến hành phân tích chuỗi XML, tải lại các dll cần thiết từ server, tạo lại các control và hiệu ứng, đăng ký lại các sự kiện.
4. Sau khi hoàn tất bước 2, ứng dụng đã sẵn sàng được sử dụng. Trong quá trình hoạt động:
   * Hệ thống sẽ giám sát ứng dụng, khi một control nào đó phát sinh ra sự kiện, hệ thống sẽ tiến hành phân phối sự kiến đó đến các control có nhu cầu xử lý sự kiện đó.
   * Control có thể có nhu cầu cần dữ liệu, khi ấy hệ thống sẽ gọi Service Server để lấy dữ liệu về.
5. Hệ thống sẽ hoạt động cho đến khi nào người dùng đóng ứng dụng. Khi ấy qui trình sẽ được kết thúc.

## Kiến trúc

Phân hệ được xây dựa trên nền công nghệ Silverlight 4 nhằm đảm bảo tính tương thích cao nhất với ứng dụng Silverlight được yêu cầu. Sau đây là mô hình và mô tả chi tiết các thành phần trong phân hệ này.



Hình ‑ Kiến trúc phân hệ trình diễn

Nhiệm vụ chính của phân hệ này là trình diễn lại ứng dụng mà người sử dụng yêu cầu.

|  |  |
| --- | --- |
| **Thành phần** | **Mô tả** |
| **Controls** | Các control được tạo ra trong ứng dụng được yêu cầu, dùng để tương tác với người sử dụng. |
| **Component Dowloader** | Thành phần được dùng để tải những dll cần thiết từ Web server về hệ thống. |
| **Effects** | Các hiệu ứng được gán cho các control được tạo nhằm tạo ra một giao diện sinh động, đẹp mắt |
| **Event Manager** | Đây là thành phần theo dõi các sự kiện được phát sinh từ các control. Ngoài ra, thành phần này có còn nhiệm vụ phân phối sự kiện cho các control. |
| **Present Core** | Đây là thành phần để gắng kết các thành phần khác lại với nhau. |
| **Wcf service** | Thành phần cung cấp các dịch vụ để giao tiếp với cơ sở dữ liệu của hệ thống. |

Bảng ‑ Các thành phần của phân hệ trình diễn

## Tính năng của hệ thống

Phân hệ này là một ứng dụng web có tính năng chính là trình diễn lại ứng dụng Silverlight đã được người dùng thiết kế.

Để sử dụng người dùng chỉ cần đưa đường dẫn đến trang chứa ứng dụng web này với tham số của trang web đó là định danh của ứng dụng Silverlight muốn trình diễn lại. Sau đó, hệ thống sẽ hiển bảng thông báo với người dùng là hãy chờ cho ứng dụng được tải hoàn tất. Sau khi ứng dụng được tải và dựng lại xong, người dùng đã có thể bắt đầu sử dụng ứng dụng.

Ngoài ra, hệ thống này còn hai tính năng được hoạt động ngầm đó là tính năng quản lý ứng dụng sau khi được dựng lại và tính năng load dữ liệu từ một nguồn dữ liệu cho trước.

Với tính năng quản lý ứng dụng sau khi được dựng lại, hệ thống lưu trữ những thông tin về các gắng kết sự kiện giữa các control. Khi một control phát sinh sự kiện, hệ thống sẽ tiến hành phân phối đến sự kiện cho các control có nhu cầu xử lý thông qua việc gọi hàm của control đó.

Với tính năng load dữ liệu từ nguồn dữ liệu cho trước, khi ứng dụng hoạt động có thể control DataList sẽ cần lấy dữ liệu từ một nguồn nào đó để thể hiện cho người dùng. Khi đó, hệ thống sẽ gửi yêu cầu đến cho Service Server để lấy dữ liệu được yêu cầu. Sau khi đã lấy dữ liệu về, hệ thống sẽ tiến hành liên kết dữ liệu và hiển thị dữ liệu được lấy về lên.

## Kết luận

Trong chương này đã trình bày về các qui trình, kiến trúc, tính năng cũng như cách sử dụng của phân hệ trình diễn. Phân hệ này được triển khai trền nên web với công nghệ nền tảng mới (Silverlight 4) để có thể trình diễn lại một cách chính xác ứng dụng mà người sử dụng yêu cầu như khi đã thiết kế.

# Kết luận

*✍ Nội dung của chương này trình bày các kết quả đạt được và hướng phát triển của đề tài.*

## Các kết quả đạt được

Dựa vào khảo sát các hệ thống Mashup hiện có trên thế giới bao gồm: **iGoogle, Dapper…** và các công cụ để thiết kế giao diện cho ứng dụng web hiện này trên thế giới bao gồm **Wix**, **MoonFruit**, **SWiSH Max 3**, **Microsoft Expression Blend 4**…, nhóm chúng em đã xác định các yêu cầu và tìm hiểu các kỹ thuật để xây dựng công cụ hỗ trợ thiết kế giao diên, cùng với tìm hiểu các kỹ thuật để phát triển ứng dụng với **Silverlight 4** và công nghệ **.NET Framework** **cho Silverlight**, nhóm đã xây dựng thành công *hệ thống công cụ hỗ trợ thiết kế trực quan giao diện ứng dụng Silverlight từ hệ thống website đã có bằng công nghệ Silverlight*.

Để mình họa các tính năng chính của hệ thống, nhóm đã sử dụng công cụ hỗ trợ thiết kế do nhóm xây dựng để thiết kế các ứng dụng bao gồm *ứng dụng xem tin tức tổng hợp*, *ứng dụng xem phim, ứng dụng bán hàng*, …

Với việc đề xuất các qui trình và giải pháp cho hệ thống công cụ cùng với quá trình nghiên cứu các kỹ thuật và công nghệ lập trình hiện đại của Microsoft: **Silverlight 4**, kỹ thuật **Reflection** trong **.NET Framework**, các open source bao gồm **PropertyGrid**, **MapulRibbon**, nhóm đã xây dựng hệ thống bao gồm:

* Phân hệ công cụ hỗ trợ thiết kế trực quan ứng dụng Silverlight được xây dựng bằng công nghệ **Silverlight 4**. Các tính năng chính bao gồm:
  + Dễ dàng nhanh chóng tạo ra ứng dụng Silverlight, hỗ trợ việc thiết kế trực quan bằng các thao tác drag drop, hỗ trợ các control menu, list… và các hiệu ứng cho control, list… Đồng thời, người dùng có thể liên kết dữ liệu từ nguồn là chuỗi xml với list và phối hợp xử lý giữa các control.
  + Xem trước (preview) ứng dụng đã được thiết kế.
* Phân hệ trình diễn lại ứng dụng Silverlight đã được thiết kế, được xây dựng bằng công nghệ **Silverlight 4**. Các tính năng chính bao gồm:
  + Trình diễn lại ứng dụng Silverlight đã được người dùng thiết kế.

## Hướng phát triển của đề tài

Hiện nay, các công nghệ RIA đã dần được sử dụng nhiều hơn, tuy nhiên vẫn còn một số rất lớn số website được xây dựng theo công nghệ truyền thống. Do đó đã xuất hiện một nhu cầu xây dựng nhanh một ứng dụng RIA từ hệ thống website đã có, khóa luận được phát triển nhằm phục vụ nhu cầu đó. Tuy nhiên, do sự hạn chế về mặt công nghệ, cùng với sự giới hạn về thời gian thực hiện khóa luận nên trong quá trình khảo sát và phát triển thống vẫn còn nhiều thiếu sót. Sau đây là những dự định phát triển tiếp theo của đề tài mà chúng em đề xuất:

* Hoàn thiện các tính năng đang có của hệ thống, giúp tạo ra sự dễ dùng hơn đó với công cụ.
* Bổ sung thêm các control và hiệu ứng cho hệ thống nhằm tạo sự đa dạng.
* Hỗ trợ tạo ra hiệu ứng mới từ những hiệu ứng đang có ngay trên giao diện thiết kế.
* Hỗ trợ việc viết mã nguồn để điều khiển các thành phần của ứng dụng.
* Hỗ trợ việc kết xuất ra tập tin để người dùng có thể đem đi triển khai ở nơi khác.
* Hỗ trợ xây dựng ra ứng dụng trên những công nghệ RIA, hiện nay mới chỉ hỗ trợ xây dựng Silverlight.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Jesse James Garrett, *AJAX: A new Approach to web Applications ,* Adaptive Path (2/2005). <http://www.adaptivepath.com/ideas/essays/archives/000385.php>

[] Lori A. MacVittie, *XAML in a Nutshell*,O'Reilly Media, Inc (2006)

[3] Macromedia, *Macromedia Flash MX—A next-generation rich client* (3/2002) [www.adobe.com/devnet/flash/whitepapers/richclient.pdf](http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf).

[4] Matthew MacDonald, *Pro Silverlight 3 in C*, Apress (2009)

[5] Ngô Bá Nam Phương, *Nghiên cứu việc biểu diễn và tổng hợp Widget phục vụ Mashup*, Luận văn Thạc sĩ Khoa học Máy tính, Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM (5/2010).

[6] Robert Lair, *Beginning Silverlight 3*, Apress (2009)

[7] Jin Yu, Boualem Benatallah, Regis Saint-Paul, Fabio Casati, Florian Daniel, Maristella Matera, *A Framework for Rapid Integration of Presentation Components*. In: Proceedings of the 16th World Wide Web Conference WWW 2007 (2007), tr. 923–932.

[8] Cokkiy, *Silverlight Carousel: Creating a Silverlight Control Displays Picture in an Interactive Carousel* (5/2009). <http://www.codeproject.com/KB/silverlight/carousel.aspx>

[9] Darick C., *CoverFlow*, <http://silverlightcoverflow.codeplex.com/> (truy cập 7/2010)

[10] Marcus Cake, *Web 1.0, Web 2.0, Web 3.0 and Web 4.0 explained*, <http://www.marcuscake.com/key-concepts/internet-evolution> (truy cập 7/2010)

[11] Hyspdrt, *PropertyGrid*, <http://slg30.codeplex.com/> (Release 2, 8/2009)

[12] Jamil Hallal, *Silverlight – Html Host Control* <http://www.codeproject.com/Tips/74491/Silverlight-Html-Host-Control.aspx> (5/2010)

[13] Mark Betz, *Gradient Maker*, <http://www.markbetz.net/sl/gradient>/ (5/2010)

[14] MSDN online, Microsoft, *Network Security Access Restrictions in Silverlight*, <http://msdn.microsoft.com/en-us/library/cc645032%28VS.95%29.aspx> (5/2010)

[15] MSDN online, Microsoft, *Reflection (C# Programming Guide)*, <http://msdn.microsoft.com/en-us/library/ms173183%28VS.80%29.aspx> (5/2010)

[16] Peter Bromberg, *Silverlight: Handling Cross-Domain Images and Gifs*, <http://www.eggheadcafe.com/tutorials/aspnet/c0046ba1-5df5-486a-8145-6b76a40ea43d/silverlight-handling-cro.aspx> (5/2010)

[17] Saltanov Maxim, *Ribbon for Silverlight*, <http://slribbon.codeplex.com/> (5/2010)

Phụ lục A – Ví dụ về việc kế thừa và cài đặt hiệu ứng

Dưới đây là mã nguồn của hiệu ứng *WipeIn* được kế thừa từ *BasicEffect*

using System;

using System.Net;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

using System.Windows.Ink;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Shapes;

using BasicLibrary;

namespace EffectLibrary

{

public class WipeIn : BasicEffect

{

// vi tri bat dau thuc hien hieu ung Wipe

public enum BeginWipe

{

TOP,

BOTTOM,

LEFT,

RIGHT,

TOPLEFT,

TOPRIGHT,

BOTTOMLEFT,

BOTTOMRIGHT

}

#region attributes

private BeginWipe beginPos = BeginWipe.TOP;

private Storyboard sb;

private TimeSpan duration = TimeSpan.FromMilliseconds(500);

private TimeSpan beginTime = TimeSpan.FromMilliseconds(0);

double width, height;

private Rectangle rectangle;

private GradientStop gs1, gs2;

private LinearGradientBrush brush;

#endregion attributes

#region properties

// khai bao cac property

public BeginWipe BeginPos

{

get { return beginPos; }

set

{

beginPos = value;

UpdateRectangle();

}

}

public double Duration

{

get { return duration.TotalMilliseconds; }

set

{

duration = TimeSpan.FromMilliseconds(value);

InitStoryboard();

}

}

public double BeginTime

{

get { return beginTime.TotalMilliseconds; }

set

{

beginTime = TimeSpan.FromMilliseconds(value);

InitStoryboard();

}

}

#endregion properties

public WipeIn(EffectableControl control)

: base(control)

{

// dang ky cac property

parameterNameList.Add("BeginPos");

parameterNameList.Add("Duration");

parameterNameList.Add("BeginTime");

// tao ra vat lieu la rectangle phu len be mat cua control

rectangle = new Rectangle();

rectangle.Width = width;

rectangle.Height = height;

Canvas.SetZIndex(rectangle, 63000);

Canvas.SetTop(rectangle, 0);

Canvas.SetTop(rectangle, 0);

control.CanvasRoot.Children.Add(rectangle);

// khoi tao brush de to cho vat lieu

brush = new LinearGradientBrush();

gs1 = new GradientStop();

gs2 = new GradientStop();

gs1.Color = Colors.Black;

gs1.Offset = 0;

gs2.Color = Color.FromArgb(0, 0, 0, 0);

gs2.Offset = 0;

brush.GradientStops.Add(gs1);

brush.GradientStops.Add(gs2);

rectangle.Fill = brush;

UpdateRectangle();

// goi ham khoi tao hieu ung

InitStoryboard();

control.SizeChanged += new SizeChangedEventHandler(control\_SizeChanged);

}

void control\_SizeChanged(object sender, SizeChangedEventArgs e)

{

// goi ham cap nhat lai hinh chu nhat khi control co thay doi kich thuoc

width = e.NewSize.Width;

height = e.NewSize.Height;

UpdateRectangle();

}

private void InitStoryboard()

{

// khoi tao story board de thuc hien hieu ung

TimeSpan ts = TimeSpan.FromMilliseconds(duration.TotalMilliseconds \* 0.4 / 1.4);

sb = new Storyboard();

sb.Completed += new EventHandler(sb\_Completed);

DoubleAnimation doubleAnimation1 = new DoubleAnimation() { BeginTime = beginTime, Duration = duration.Subtract(ts), From = 1, To = 0 };

Storyboard.SetTarget(doubleAnimation1, gs1);

Storyboard.SetTargetProperty(doubleAnimation1, new PropertyPath("Offset"));

sb.Children.Add(doubleAnimation1);

DoubleAnimation doubleAnimation2 = new DoubleAnimation() { BeginTime = ts + beginTime, Duration = duration.Subtract(ts), From = 1, To = 0 };

Storyboard.SetTarget(doubleAnimation2, gs2);

Storyboard.SetTargetProperty(doubleAnimation2, new PropertyPath("Offset"));

sb.Children.Add(doubleAnimation2);

}

void sb\_Completed(object sender, EventArgs e)

{

// phat sinh ra su kien hieu ung da thuc hien hoan tat

base.RaiseEffectCompleteEvent(this);

}

private void UpdateRectangle()

{

// cap nhat lai vi tri va kich thuoc cua hinh chu nhat

rectangle.Width = width;

rectangle.Height = height;

Point start, end;

switch (beginPos)

{

case BeginWipe.TOP:

start = new Point(0.5, 1);

end = new Point(0.5, 0);

break;

case BeginWipe.BOTTOM:

start = new Point(0.5, 0);

end = new Point(0.5, 1);

break;

case BeginWipe.LEFT:

start = new Point(1, 0.5);

end = new Point(0, 0.5);

break;

case BeginWipe.RIGHT:

start = new Point(0, 0.5);

end = new Point(1, 0.5);

break;

case BeginWipe.TOPLEFT:

start = new Point(1, 1);

end = new Point(0, 0);

break;

case BeginWipe.TOPRIGHT:

start = new Point(0, 1);

end = new Point(1, 0);

break;

case BeginWipe.BOTTOMLEFT:

start = new Point(1, 0);

end = new Point(0, 1);

break;

case BeginWipe.BOTTOMRIGHT:

start = new Point(0, 0);

end = new Point(1, 1);

break;

default:

start = end = new Point(0, 0);

break;

}

brush.StartPoint = start;

brush.EndPoint = end;

}

#region override methods

// override lai cac ham Start, Stop, DetachEffect, SetSelHandle

public override void Start()

{

gs1.Offset = 1;

gs2.Offset = 1;

sb.Begin();

}

public override void Stop()

{

sb.Stop();

}

public override void DetachEffect()

{

Canvas.SetLeft(control.Control, 0);

Canvas.SetTop(control.Control, 0);

control.CanvasRoot.Children.Remove(rectangle);

}

protected override void SetSelfHandle()

{

}

#endregion override methods

}

}

Nội dung thêm vào tập tin info.xml:

<Control>

<Name>EffectLibrary.WipeIn</Name>

<DllFilename>EffectLibrary.dll</DllFilename>

<DllReferences>

</DllReferences>

<DisplayName>Wipe In</DisplayName>

<Description>Wipe In effect</Description>

<IconName>WipeIn.png</IconName>

<Group>AppearEffect</Group>

</Control>