# TÓM TẮT LUẬN VĂN

**ĐỀ TÀI KHÓA LUẬN TỐT NGHIỆP**

**Tên đề tài:** XÂY DỰNG HỆ THỐNG QUẢN LÝ, ĐẶT MÓN ĂN VÀ QUẢNG CÁO SẢN PHẨM TRONG NHÀ HÀNG DỰA TRÊN CÔNG NGHỆ ANDROID – LIFERAY PORTAL

1. **Các vấn đề nghiên cứu**

Đề tài được thực hiện với mục tiêu tin học hóa quy trinh quản lý và đặt các món ăn trong nhà hàng để giảm chi phí thuê nhân viên phục vụ. Đặc biệt hệ thống cung cấp chức năng quảng cáo, cho phép nhà hàng nhận thêm yêu cầu quảng cáo sản phẩm từ các công ty khác để tăng thêm lợi nhuận.

Để có thể thực hiện trên 02 công nghệ đang phổ biến hiện này là lập trình trên thiết bị di động Android và Liferay Portal, đề tài phải nghiên cứu một số vấn đề sau:

* **Android**
  + Thiết lập môi trường làm việc, công cụ phát triển và cấu hình máy ảo Android.
  + Sử dụng các đơn vị quản lý giao diện trong Android: Activity, Fragment và Intent.
  + Nghiên cứu các thành phần giao diện người dùng trên Android (Android User Interface).
  + Xây dựng giao diện sử dụng các đối tượng View trong thư viện Android.
  + Xử lý và hiển thị hình ảnh.
  + Sử dụng các loại Menu View.
  + Hiển thị thông tin dưới dạng danh sách: ListView và dạng lưới: GridView.
  + Lưu trữ dữ liệu bền vững trong Android.
  + Xử lý tác vụ bất đồng bộ: AsynTask, bộ định thời: Timer và đối tượng xử lý: Handler.
  + Gửi thông điệp và Email.
  + Truy xuất dữ liệu từ Web Service sử dụng định dạng SOAP và thư viện KSOAP2.
  + Đồng bộ hóa dữ liệu giữa các máy trạm Android và máy chủ.
* **Liferay Portal** 
  + Cài đặt và cấu hình Liferay Portal.
  + Xây dựng Website bằng JSF và thư viện PrimeFace.
  + Tạo, quản lý và lưu trữ dữ liệu sử dụng Service Builder.
  + Cung cấp Web Service bằng công nghệ Axis của Apache.

1. **Một số phương pháp tiếp cận:**

Để tiến hành nghiên cứu và xây dựng cơ sở cho công trình chúng tôi đã sử dụng một số phương pháp luận nghiên cứu dưới đây:

* + - * **Phương pháp quan sát khoa học**

Quan sát trực tiếp các hoạt động quản lý và đặt món ăn tại quán ăn gia đình Đông Hải (Khu phố 11 phường Hố Nai, Biên Hòa, Đồng Nai) và nhà hàng Tera (Tầng T, tòa nhà FTOWN khu công nghệ cao, quận 9 thành phố Hồ Chí Minh) để tìm ra các vấn đề đang tồn tại và kiểm chứng các lý thuyết về công nghệ đã có nhằm tìm ra hướng giải quyết và phát triển đề tài.

* **Phương pháp phân tích và tổng hợp lý thuyết**

Từ những chức năng và yêu cầu đã được đặt ra, đọc và phần tích các điểm nỗi bật của các kiến thức, công nghệ có liên quan đến những mặt của công trình sau đó rút ra những điểm ưu và nhược của công nghệ đó. Tổng hợp các ưu nhược điểm để quyết định sẽ sử dụng công nghệ nào để giải quyết các vấn đề trong đề tài.

* **Phương pháp phân loại lý thuyết**

Nghiên cứu và sắp xếp các kiến thức lý thuyết đã có và các công cụ thực hiện thành các nhóm để sử dụng trong quá trình thực hiện công trình.

* **Phương pháp mô hình hóa**

Xác định các đối tượng thực tế được nghiên cứu trong công trình, mô hình hóa các đối tượng này dưới dạng tri thức máy để thao tác và tạo cơ sở dữ liệu.

* **Phương pháp thực nghiệm và đặt giả thiết**

Tiến hành thực hiện công trình song song với việc kiểm tra các công việc thực tế của nhà hàng để đảm bảo tính khả thi của công trình. Liên tục đặt ra các giả thiết để mở rộng và hoàn thiện công trình.

**3. Kết quả chính đạt được**

* Ứng dụng iRestAds trên Android cho phép thực khách đặt món ăn thông qua máy tính bảng và hiển thị các quảng cáo được nhà hàng nhận từ những công ty khác.
* Website được xây dựng trên công nghệ Liferay Portal để quản lý các món ăn, thực đơn và quảng cáo cho nhà hàng.

# MỞ ĐẦU

## Hiện trạng và khó khăn thực tế

Hiện nay số lượng nhà hàng chuyên cung cấp dịch vụ ăn uống xuất hiện nhiều và phát triển một cách mạnh mẽ. Hầu hết các nhà hàng đều đang thực hiện kinh doanh theo hướng tin học hóa bằng cách sử dụng các phần mềm quản lý. Tuy nhiên các phần mềm này chưa thực sự mạnh mẽ vì chỉ thực hiện được việc quản lý một lượng món ăn nhất định và tính tiền cho thực khách. Mặc dù đã áp dụng CNTT vào hoạt động của nhà hàng nhưng vẫn còn nhiều khó khăn như chưa giảm được chi phí thuê nhân viên để nhận yêu cầu gọi món từ thực khách, sự không hài lòng của thực khách khi phải chờ nhân viên đến quá lâu hay nhà hàng phải tốn kém tiền bạc để thay đổi thực đơn hoặc phải giải thích với khách hàng khi không thể cung cấp được món ăn mà khách gọi.

Bên cạnh những vấn đề về quản lý món ăn, thực đơn và đặt món ăn. Các nhà hàng hiện nay còn có xu hướng nhận quảng cáo sản phầm cho các công ty khác hoặc tự quảng bá cho những sự kiện, món ăn mới của nhà hàng. Việc quảng cáo này được thực hiện dưới hình thức tờ rơi hay banner quảng cáo, phương thức quảng cáo này đã không còn nhiều hiệu quả và bị hạn chế số lượng do không gian.

## Lý do chọn đề tài

Trước những hiện trạng và khó khắn đã trình bày ở trên, chúng tôi quyết định thực hiện đề tài xây dựng hệ thống quản lý, đặt món ăn và quảng cáo sản phẩm trong nhà hàng nhằm phát triển một ứng dụng giúp tiết kiệm tiền bạc và nhân lực phục vụ thực khách cho nhà hàng, đem lại sự mới mẻ và thú vị cho thực khách khi đến nhà hàng. Đồng thời xây dựng một phương pháp quảng cáo sản phẩm mới với mong muốn nâng cao hiệu quả, số lượng và sự tương tác của khách hàng.

Về công nghệ, thời gian gần đây việc phát triển về thị phần của các dòng điện thoại thông minh ngày càng tăng đặc biệt là Android với 50.9% thị phần (tính đến tháng 6 năm 2012 theo comScore ) kéo theo sự phát triển về các công cụ lập trình và thư viện giúp việc thực hiện việc phát triển các ứng dụng trên thiết bị di động dễ dàng, đa dạng. Việc thực hiện để tài trên hệ điều hành Android sẽ làm tăng khả năng ứng dụng thực tế và khả năng mở rộng của đề tài.

## Mục tiêu đề tài

Xây dựng một hệ thống có khả năng quản lý thông tin các món ăn, thực đơn, hóa đơn của nhà hàng, có khả năng thực hiện các nghiệp vụ một cách nhanh chóng và chính xác, cung cấp các chức năng cho thực khách và nhân viên nhà hàng có thể tương tác với nhau từ xa.

* + - * Nhân viên :
        + Quản lý các món ăn dưới dạng kho món ăn.
        + Quản lý các thực đơn theo từng ngày.
        + Nhận yêu cầu đặt món của khách hàng mà không cần trực tiếp đến bàn.
        + Cập nhật, thông báo trạng thái của yêu cầu đặt món.
        + Quản lý hóa đơn và thanh toán.
        + Nhận các đơn quảng cáo sản phẩm từ công ty khác.
      * Thực khách :
        + Xem và đặt các món ăn thông qua máy tính bảng Android.
        + Kiểm tra, theo dõi tiến độ thực hiện các món ăn mình đã đặt.
        + Xem thông tin về hóa đơn, số tiền mình đã sử dụng khi đặt món.
        + Xem và liên hệ với các nhà cung cấp sản phẩm khi xem quảng cáo tại nhà hàng.

## Phạm vi và nội dung nghiên cứu

Đề tài nghiên cứu xoay quanh các kỹ thuật lập trình trên nền tảng hệ điều hành Android nhằm đáp ứng được các nhu cầu về quy trình phục vụ thực khách, giúp một nhà hàng có thể hoạt động theo hướng tin học hóa, tiết kiệm được thời gian, tiền bạc và nhân lực. Đặc biệt cung cấp một số chức năng về quảng cáo sản phầm trong nhà hàng.

Ngoài kỹ thuật lập tình Android, đề tài còn sử dụng giải pháp công thông tin Liferay Portal để xây dựng một Website cho phép nhân viên quản lý thông tin về món ăn và sản phẩm quảng cáo của nhà hàng. Liferay Portal cũng được sử dụng trong việc cung cấp các dịch vụ Web (hay còn gọi là Web Service) cho các máy trạm Android.

**Nội dung nghiên cứu**

Đề tài chủ yếu nghiên cứu về các nội dung của lập trình trên hệ điều hành Android, bao gồm các nội dung sau :

**ANDROID**

* + - * **Phần 1: Những điều cần thiết để bắt đầu lập trình trên Android**

Trình bày những nội dung căn bản của hệ điều hành Android và một số thông tin hiện tại về hệ điều hành này. Một số chức năng của những thiết bị Android, cách thức tải và cài đặt những công cụ cần thiết để phát triển ứng dụng Android cũng như làm thế nào để chạy các ứng dụng trên máy ảo Android (Android Emulator).

* + - * **Phần 2 : Các thành phần quản lý giao diện Activities, Fragments, và Intent**

Làm quen với 03 khái niệm căn bản trong lập trình Android. Activities và Fragments được hiểu là những khối xây dựng giao diện căn bản trong ứng dụng Android. Phần này diễn giải cách thức liên kết các Activities lại với nhau sử dụng Intent, một thành phần rất đặc biệt và chỉ có ở hệ điều hành Android và tìm hiểu thêm về một số Intent Filter thường dùng.

* + - * **Phần 3 : Các thành phần giao diện người dùng của Android**

Trình bày về những thành phần khác nhau tạo lên giao diện người dùng cho ứng dụng Android. Nghiên cứu về các loại **Layout** khác nhau có thể được dùng để xây dựng giao diện cho ứng dụng, và một số loại sự kiện liên quan đến sự tương tác với người dùng.

* + - * **Phần 4 : Thiết kế giao diện người dùng sử dụng đối tượng View**

Thông qua những đối tượng View căn bản chúng ta sẽ học để xây dựng giao diện ứng dụng Android. Làm quen với 03 nhóm giao diện chính: Basic View, Picker View và List View. Đặc biệt, tìm hiểm về các sử dụng các Fragment chuyên dụng trong Android 3.0 và 4.0.

* + - * **Phần 5 : Xử lý, hiển thị hình ảnh và sử dụng Menu**

Tiếp tục giải thích về các đối tượng View. Ở phần này sẽ trình bày về cách thức hiển thị hình ảnh thông qua các đối tượng View khác nhau, cũng như làm thế nào để tạo Option Menu và Context Menu trong một ứng dụng.

* + - * **Phần 6 : AsynTask, Timer và Handler**

Giới thiệu về xử lý các tác vụ bất đồng bộ, thường được dùng trong cập nhật giao diện trong thời gian chạy chương trình (run time) hay trao đổi dữ liệu với Server. Bộ định thời Timer và cách sử dụng đối tượng xử lý Handler.

* + - * **Phần 7 : Lưu trữ dữ liệu**

Nghiên cứu cách thức để lưu trữ dữ liệu trong ứng dụng Anroid. Tìm hiểu về những kỹ thuật khác nhau trong việc lưu trữ liệu thông qua việc thao tác trên các tập tin, lưu trữ chúng bằng bộ nhớ trong và bộ nhớ ngoài (Thẻ SD). Đặc biệt là nghiên cứu về cách thức tạo và sử dụng cơ sở dữ liệu SQLite trong ứng dụng Android.

* + - * **Phần 8 : Gửi thông điệp**

Nghiên cứu về 02 cách thức được quan tâm nhiều nhất rong lập trình trên thiết bị di động là gửi tin nhắn SMS và Email. Cách thức để gửi và nhận tin nhắn SMS cũng như Email bằng phương pháp lập trình.

* + - * **Phần 9 : Tương tác và truy xuất dữ liệu từ Web Service**

Trình bày những vấn đề về kết nối, gọi hàm từ ứng dụng Android đến một Web Service bên ngoài sử dụng thư viện KSOAP2 cũng như làm thế nào để xử lý các dữ liệu trả về.

**LIFERAY PORTAL**

* **Phần 1 : Cài đặt và cấu hình Liferay Portal.**

Trình bày về cài đặt, cấu hình các biến môi trường và một số thiết lập căn bản để sử dụng công thông tin Liferay Portal

* **Phần 2 : Xây dựng Website bằng JSF và thư viện PrimeFace.**

Cách thức xây dựng các thành phần giao diện, những sự kiện tương tác với người dùng và xử lý tương tác người dùng sử dụng thư viện PrimeFace.

* **Phẩn 3 :** **Tạo, quản lý và lưu trữ dữ liệu sử dụng Service Builder.**

Mô hình hóa các đối tượng, lưu trữ và thao tác chúng sử dụng Service Builder được Liferay hỗ trợ.

* **Phần 4 : Thay đổi và điều chỉnh những truy vấn cơ sở dữ liệu**
* **Phần 5 :** **Cung cấp Web Service bằng công nghệ Axis của Apache.**

# NỘI DUNG NGHIÊN CỨU : ANDROID

Trong chương này chúng tôi sẽ trình bày về những nghiên cứu cần thiết và căn bản nhất để lập trình được trên thiết bị di động nền tảng Android. Vì là kiến thức căn bản nên những ví dụ minh họa được đề cấp đến được trích từ cuốn sách nổi tiếng về lập trình Android : Beginning Android 4 Application Development của Wei-Meng Lee do nhà xuất bản Wrox xuất bản năm 2012. Các ví dụ này đã được đơn giản hóa nhằm đạt được hiệu quả trong việc chuyền đạt. Với mục tiêu sẽ trình bày được lý thuyết về những kỹ thuật lập trình trên thiết bị di động mà chúng tôi nghiên cứu được, thứ tự các phần trong chương được sắp xếp theo độ khó tăng dần giúp dễ dàng xây dựng và tóm lược những kiến thức cần thiết.

Tuy nhiên lập trình trên Android là một chủ đề lớn và trong phạm vi của đề tài này, chúng tôi chỉ có thể trình bày những nội dung mà theo chúng tôi là cần thiết cho những người bắt đầu tiếp cận với lập trình Android có thể phát triển được ứng dụng. Chương này có 03 mục đích chính :

* Giới thiệu về lập trình Android và những kiến thức căn bản.
* Các bước thực hiện để xây dựng một ứng dụng Android.
* Xử lý trao đổi thông tin từ ứng dụng Android đến dịch vụ Web

## Những điều cần thiết để bắt đầu lập trình trên hệ điều hành Android

Nội dung chính

* Khái niệm về Android.
* Những phiên bản đã được tung ra thị trường.
* Tải và cài đặt những công cụ phát triển và SDK cho việc phát triển ứng dụng Android.
* Viết ứng dụng Android đầu tiên : Hello world

### Khái niệm về Android

**Android là gì ?**

Android là một hệ điều hành trên thiết bị di động, được xây dựng trên nền tảng hệ điều hành Linux, nó được thiết kế riêng cho những thiết bị di động có màn hình cảm ứng (touchscreen) như điện thoại thông minh và máy tính bảng. Ban đầu được phát triển bởi công ty Android, Inc. Vào nằm 2005 trong chiến dịch tham gia vào thị trường thiết bị di động của mình, Google đã mua lại Android, dành lấy quyền phát triển cũng như đội ngũ phát triển hệ điều hành Android. Google muốn Android trở thành một hệ điều hành mở và miễn phí, do đó hầu hết mã lệnh của Android đều được công khai với những bản quyền về mã nguồn mở của Apache. Điều này có nghĩa là bất cứ ai muôn sử dụng Android có thể tải về toàn bộ mã nguồn của Android. Hơn nữa những nhà sản xuất thiết bị cũng có thể thêm vào những tính năng mở rộng của riêng họ cho Android và chế biến lại để tạo ra sự khác biết cho sản phẩm của họ.

Quy trình phát triển đơn giản của Android rất hấp dẫn và được quan tâm nhiều bởi các hãng sản xuất thiết bị di động. Thành công của Android đặc biệt có ích cho những công ty bị ảnh hưởng bởi hiện tượng iPhone của hãng Apple, một sản phẩm siêu thành công trong ngành công nghiệp điện thoại thông minh. Các công ty đó gồm có Motorola và Sony Ericsson, những công ty này đã bỏ nhiều năm để phát triển hệ điều hành riêng của mình nhưng khi iPhone được bày bán thì nhiều nhà sản xuất đã phải tìm một hướng đi mới để phục hồi các sản phẩm của họ. Những nhà sản xuất này nhận thấy Android là một giải pháp tốt, họ tiếp tục thiết kế các thiết bị phần cứng của mình và sử dụng Android làm hệ điều hành.

Lợi ích chính của việc áp dụng Android là nó cung cấp một hướng tiếp cận thống nhất cho việc phát triển ứng dụng. Những lập trình viên chỉ cần phát triển cho Android và ứng dụng của họ sẽ chạy được trên nhiều loại thiết bị sử dụng hệ điều hành Android khác nhau.

### Những phiên bản đã được tung ra thị trường và một số chức năng của Android.

Từ lần xuất hiện đầu tiên năm 2009 đến nay Android đã sản xuất một số phiên bản sau đây.

Bảng ‑ Những phiên bản của Android

|  |  |  |
| --- | --- | --- |
| **Android Version** | **Ngày xuất bản** | **Tên phiên bản** |
| **1.1** | Ngày 09 tháng 02 năm 2009 |  |
| **1.5** | Ngày 30 tháng 04 năm 2009 | Cupcake |
| **1.6** | Ngày 15 tháng 09 năm 2009 | Donut |
| **2.0 / 2.1** | Ngày 26 tháng 10 năm 2009 | Eclair |
| **2.2** | Ngày 20 tháng 05 năm 2010 | Froyo |
| **2.3** | Ngày 06 tháng 12 năm 2010 | Gingerbread |
| **3.0/3.1/3.2** | Ngày 22 tháng 02 năm 2011 | Honeycomb |
| **4.0** | Ngày 19 tháng 10 năm 2011 | Ice Cream Sandwich |

Vào tháng 02 năm 2011, Google cho ra mắt phiển bản Android 3.0, đây là phiên bản riêng dành cho máy tính bảng, hỗ trợ những thiết bị có màn hình rộng. Những ứng dụng viết trên các phiên bản trước 3.0 thì tương thích với phiên bản 3.0 và có thể chạy trên phiên bản này mà không cần chỉnh sửa gì. Tuy nhiên những ứng dụng viết trên nên 3.0 lại không thể chạy được trên các thiết bị cũ hơn. Do đó, để chắc chắn các ứng dụng trên máy tính bảng có thể chạy được trên tất cả các phiên bản vào tháng 10 năm 2011 Google cho ra mắt phiên bản 4.0 với tất cả những chức năng được giới thiệu trong phiên bản 3.0. Với đặc trưng về chức năng của đề tài nên chúng tôi quyết định sử dụng thiết bị là máy tính bảng Pop Tablet chạy trên nên phiên bản 4.0 này.

### Tải và cài đặt những công cụ phát triển và SDK cho việc phát triển ứng dụng Android.

Trước khi bắt đầu viết ứng dụng đầu tiên ta cần tải những công cụ cần thiết và bộ SDK của Android để chuẩn bị cho việc lập trình. Để phát triển ứng dụng Android ta có thể sử dụng máy tính được cài hệ điều hành MAC, Windown hoặc Linux. Tất cả những công cụ cần thiết đều miễn phí và có thể được tải về từ các trang Web. Hầu hết các ví dụ trong chương này đều có thể chạy được trên máy ảo Android. Trong báo cáo này chúng tôi sẽ trình bày trên nền Windows 8.

**Java SDK**

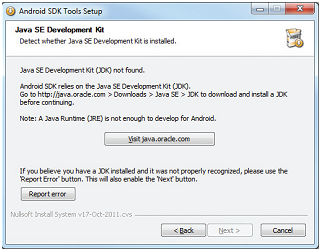
Android SDK làm việc dựa trên nền tảng của Java SE Development Kit (JDK). Do đó nếu máy tính được dùng để lập trình phải được cài đăt JDK, ta có thể tải JDK tại địa chỉ của Oracle : <http://www.oracle.com/technetwork/java/javase/downloads/index.html> và cài đặt JDK trước khi bắt đầu với những công cụ khác.

**Android SDK**

Công cụ đầu tiên và quan trọng nhất cần tải về là Android SDK. Đây là công cụ do Google cung cấp bao gồm công cụ kiểm tra lỗi (Debugger), các thư viện, máy ảo Android, tài liệu hướng dẫn, những mã nguồn đơn giản và một số bài hướng dẫn lập trình. Ta có thể tải Android SDK từ trang: <http://developer.android.com/sdk/index.html>. Sau khi tải gói Android SDK ta giải nén và được tập tin cài đặt có tên là installer\_xxx-windows.exe.

* **Cài đặt Android SDK**

Sau khi đã tải thành công tập tin installer\_xxx-windows.exe, nhấp đôi để bắt đầu tiến hành cài đặt công cụ Android. Tại màn hình đầu tiên Setup Wizard, nhấp chuột vào nút Next để tiếp tục, nếu máy tính chưa có JDK thi sẽ hiện thị thông báo lỗi như hình 2.1.3-1.

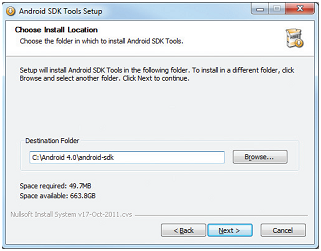


Hình ‑ Thông báo lỗi chưa cài đặt Java

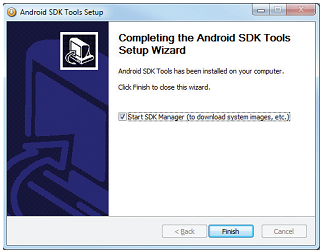
Khi đó ta cần thoát trình cài đặt và tiến hành cái JDK trước. Sau đó thực hiện lại việc cài đặt.

Tiếp theo, ta sẽ được yêu cầu cung cấp một thư mục để cài đặt công cụ Android SDK. Nhập một đường dẫn (Hình 2.1.3-2) và nhấp vào nút Next.

Sau đó ta chọn mặc định cho các đòi hỏi khác, và nhấp nút Install. Khi quá trình cài đặt hoàn tất, chọn hộp kiểm “Start SDK Manager” và nhấp vào nút Finish (Hình 2.1.3-3). Sau đó SDK Manager sẽ được khởi chạy.



Hình ‑ Chọn thư mục cài đặt



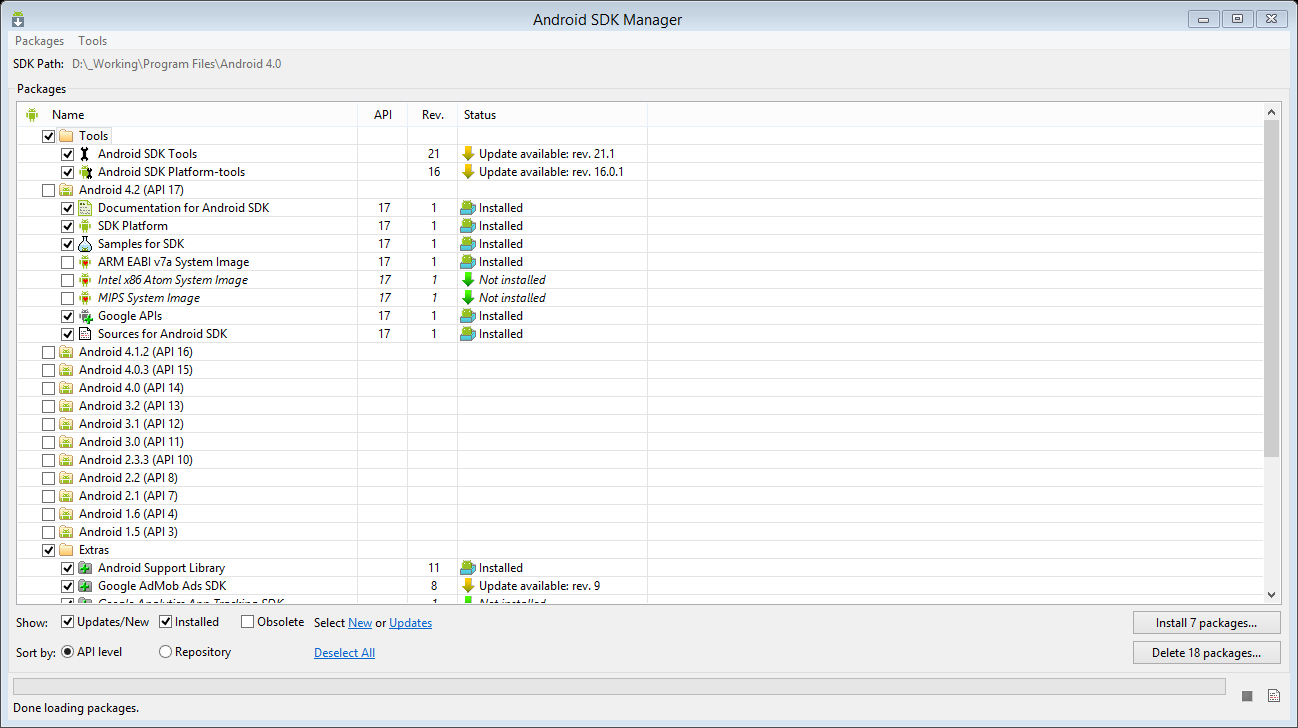
Hình ‑ Hoàn tất cài đặt Android JDK

* **Cấu hình Android SDK Manager**

Android SDK Manager quản lý các phiên bản khác nhau của Android SDK đã được cài đặt trên máy tính. Khi ứng dụng này được chạy sẽ hiển thị danh sách những mục khác nhau của các phiên bản Android SDK. Chọn những công cụ, tài liệu hướng dẫn và các platform có liên quan cần cho dự án ứng dụng Android. Khi đã chọn xong các mục mong muốn, nhấp vào nút Install để tiến hành tải các mục này. Vì những mục này được tải trực tiếp từ máy chủ của Google nên thời gian tải về thường chậm, do đó ta chỉ nên chọn những mục thật sự cần thiết để tiết kiệm thời gian.

Những mục tối thiểu phải cài đặt gồm có Tools, API và Extras như hình 2.1.3-4.

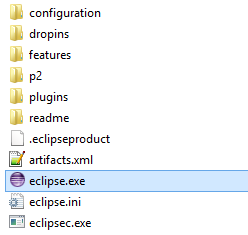
Mỗi phiên bản của hệ điều hành Android được định nghĩa bởi 1 con số biểu thị mức độ, Ví dụ Android 3.2 có mức độ là 13 (API 13), Android 4.0.3 mức độ 15 (API 15). Khi thực hiện viết báo cáo này Google đã cho xuất bản Android version 4.2 với mức độ 17. Với mỗi mức độ, Google cung cấp 2 platform là SDK Platform và Google APIs, sự khác nhau chính của 2 platform này là Google APIs có thêm những API riêng được cung cập bới Google như thư viện Google Maps hay Tranlates.



Hình ‑ Android SDK Manager

Sau khi nhấp nút Install ta sẽ được hỏi muốn cài đặt gọi nào, chọn “Accept All” và nhấp tiếp Install và SDK Manager sẽ tự động tải các gói được chọn.

**Eclipse**

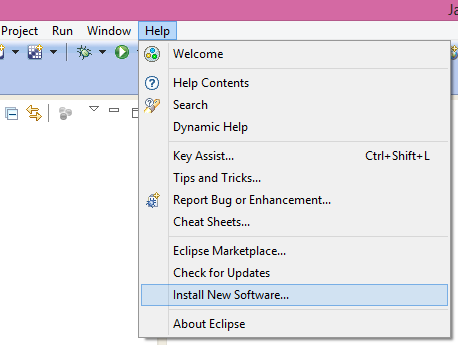
Bước tiếp theo là thiết lập môi trường tích hợp phát triển hay được gọi tắt là IDE cho việc phát triển ứng dụng Android. Hầu hết các lập trình viên Android hiện này đều sử dụng Eclipse để phát triển, đây là một môi trường phát triển phần mềm đa ngôn ngữ với hệ thống mở rộng dạng plug-in. Có thể dùng Eclipse để phát triển nhiều loại ứng dụng khác nhau sử dụng các ngôn ngữ lập trình Java, C, C++, Python và một số ngôn ngữ khác.

Hình ‑ Cấu trúc thư mục Eclipse.

Để phát triển Android, ta nên tải bản Eclipse IDE For Java EE Developers tại trang chủ của Eclipse: <http://www.eclipse.org/downloads/>. Có 6 phiên bản để lựa chọn Windows (32- and 64-bit), Mac OS X (Cocoa 32- and 64), and Linux (32- and 64-bit). Chọn phiên bản phù cần thiết cho hệ điều hành của máy tính và tải về. Khi hoàn tất tải Eclipse IDE, giải nén vào một thư mục nào đó và ta được cấu trúc thư mục như hình 2.1.3-5. Để chạy chương trình Eclipse ta nhấp đôi vào tập tin **eclipse.exe**.

* **Android Development Tools (ADT)**

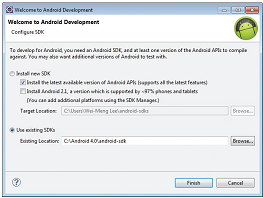
ADT là một phần mở rộng của Eclipse IDE, ADT hỗ trợ cho việc tạo và giải quyết lỗi cho ứng dụng Android. Để tiến hành cài đặt ADT ta chạy Eclipse chọn “Help” trên thanh trình đơn 🡺 chọn tiếp “Install New Software” để cài đặt plug-in Android Development Tools (ADT) cho Eclipse.



Hình ‑ Cài đặt ADT Plug-in

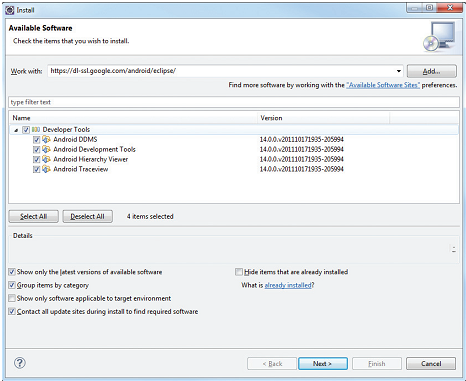
Plug-in ADT sẽ cung cấp một số chức năng mà ta có thể thực hiện trực tiếp trong Eclipse :

* + Tạo mới một dự án ứng dụng Android.
  + Sử dụng các công cụ truy xuất đến máy ảo Android và các thiết bị khác.
  + Biên dịch và giải quyết lỗi của ứng dụng Android.
  + Đóng gói ứng dụng Android thành từng gói (Android Packages APKs).

Sau khi chọn Install New Software, một cửa số được hiển thị. Nhấp địa chỉ sau vào ô Work with : <https://dl-ssl.google.com/android/eclipse/> và ấn Enter. Ta sẽ nhìn thấy những mục của Developer Tools ở phần giữa của cửa sổ. Chọn tất cả những mục được hiển thị và nhấp nút Next 2 lần (Hình 2.1.3-7).

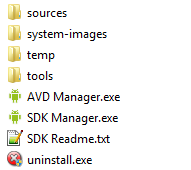
Tiếp theo, ta sẽ được yêu cầu xem và chấp nhận một số bản quyền. Nhấp chọn “I accept the terms of license agreement” và chọn “Finish”. Khi quá trình cài đặt kết thúc, ta được yêu cầu khởi động lại Eclipse. Sau khi khởi động lại ta sẽ cấu hình Plug-in đến đường dẫn Android SDK đã được tải trước đó (Hình 2.1.3-8). Chọn “Use existing SDKs” và chỉ định đến thư mục cài đặt Android SDK, nhấp “Next”. Sau bước này ta được yêu cầu gửi những thông tin đến Google. Nhấp Finish.

Hình ‑ Chỉ định Android SDK



Hình ‑ Tải Plug-in ADT trong Eclipse

**Tạo thiết bị ảo Android (Android Virtual Devices AVDs)**

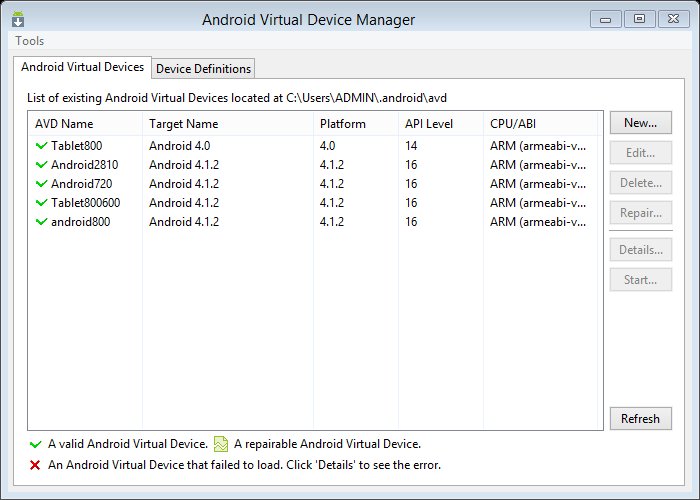
Bước tiếp theo là tiến hành tạo một thiết bị ảo Android (AVD) để dùng cho việc kiểm tra ứng dụng Android. AVD là một thể hiện giả lập cho phép mô phỏng một thiết bị thực tế. Mỗi AVD bao gồm mô phỏng phần cứng, ánh xạ đến ảnh hệ thống (System image), cùng với các mô phỏng lưu trữ ví dụ như thẻ SD (Secure Digital card).

Hình ‑ AVD Manager.exe

Ta có thể tạo ra nhiều AVD để phục vụ cho việc kiểm tra ứng dụng của mình với những cấu hình khác nhau. Việc kiểm tra này là rất quan trọng để xác nhận các hành vì của ứng dụng khi chúng chạy trên những thiết bị khác nhau. Để tạo một AVD mở thư mục cài đặt Android SDK và chạy tập tin AVD Manager.exe (Hình 2.1.3-9).

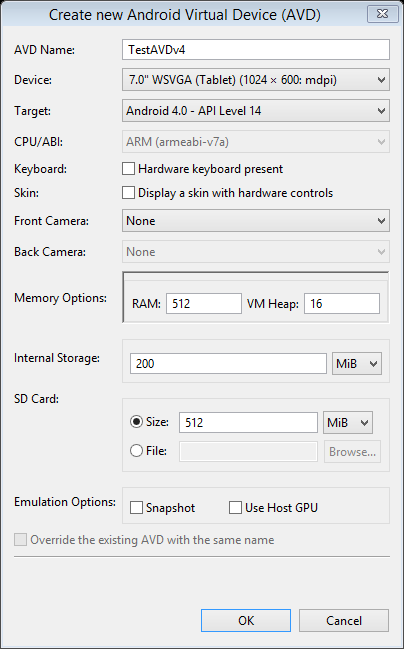
Sau khi khởi chạy, AVD Manager sẽ hiển thị tất cả những AVD đã được tạo trên máy tính. Những thông tin của máy ảo gồm có :

* + AVD Name : Tên máy ảo.
  + Target Name : Tên phiên bản Android.
  + Platform : Ký hiệu phiên bản.
  + API Level : Ký hiệu API.
  + CPU/ABI : Tên bộ xử lý.



Hình ‑ AVD Manager : Danh sách máy ảo

Để tạo một máy ảo mới ta nhấp vào nút New, khi đó cửa sổ tạo máy áo sẽ được hiển thị.

Nhập các thông tin được yêu cầu để tạo AVD.

Name: Tên máy ảo.

Device: Chọn loại máy cần mô phỏng.

Target: Chọn phiên bản Android và API tương ứng. Sau đó AVD Manager sẽ tự động chọn các thuộc tính tương thích với loại máy này trong thực tế.

Memory Options: Bộ nhớ RAM của máy.

Internal Storage: Bộ nhớ trong.

SD Card: Bộ nhớ ngoài.

Sauk hi chọn xong các thuộc tính nhấp nút OK để tạo máy ảo. Trong trường hợp của đề tài, chúng tôi tạo máy ảo tên TestAVDv4 với thiết bị là Tablet độ phân giải màn hình 1024 x 600, chạy trên nên Android 4.0 với API 14.

Sau khi đã tạo được AVD ta quay lại màn hình hiển thị danh sách các AVD đã được tạo (Hình 2.1.3-10) chọn AVD vừa tạo và nhấp nút Start. Cửa sổ Launch Options được hiện thị nhấp Launch để tiến hành chạy máy ảo.

Khi đó ta sẽ được kết quả như hình dưới.

Hình ‑ Tạo máy ảo AVD

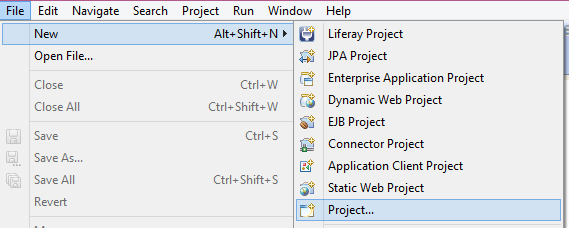


Hình ‑ Máy ảo AVD

### Tạo ứng dụng Android đầu tiên : Hello World

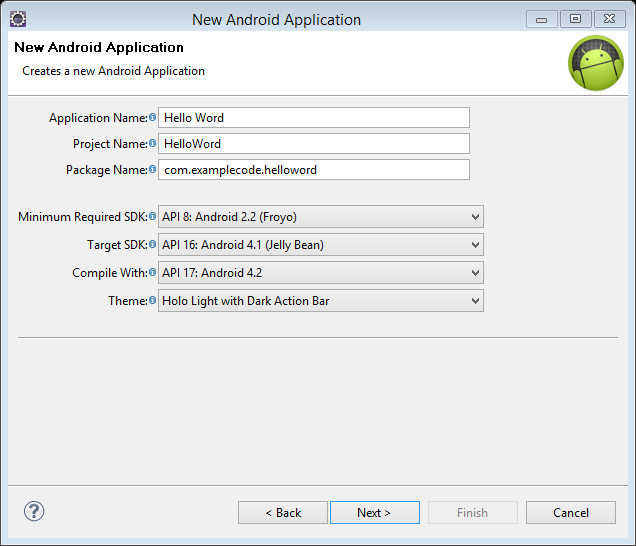
Với tất cả cácc ông cụ và SDK đã được cài đặt, ta có thể tiến hành tạo và xây dựng các ứng dụng Android. Sau đây chúng tôi xin được trình bày về cách thức tạo một ứng dụng Android sử dụng môi trường phát triển Eclipse.

**Bước 1**: Chọn File 🡺 New 🡺 Project



Hình ‑ Tạo dự án Android bằng Eclipse

**Bước 2:** Cửa sổ New Project xuất hiện, tìm đến thư mục Android 🡺 chọn Android Project Application 🡺 chọn Next để tiếp tục quá trình tạo dự án Android.

**Bước 3:** Điền giá trị cho các trường được yêu cầu. Các trường này sẽ được tự phát sinh trong tập tin Mainfish.xml sẽ được trình bày ở phần sau.

Hình ‑ Tạo dự án Android: Bước 3

Application Name: Tên của ứng dụng sẽ được hiện thi trên thiết bị.

Project Name: Tên của dự án sử dụng trong Eclipse.

Package Name: tên gói quản lý mã nguồn sẽ được lưu trong thiết bị.

Minimum Required SDK: Phiên bản SDK tối thiểu để chạy ứng dụng.

Target SDK: Phiên bản chính được chọn để phát triển ứng dụng.

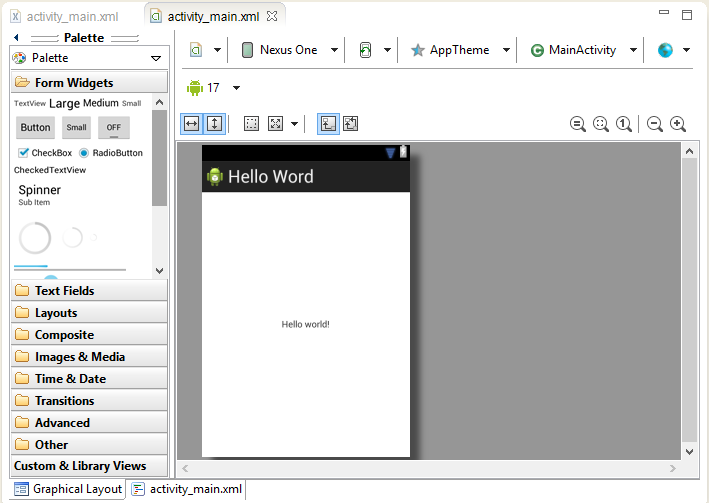
Sau khi lựa chọn xong các thông tin tiếp tục chọn Next.

**Bước 4:** Một cửa sổ được hiển thị cho phép ta chọn nơi lưu trữ mã nguồn và một vài chức năng khác. Dữ thiết lập mặc định và chọn Next

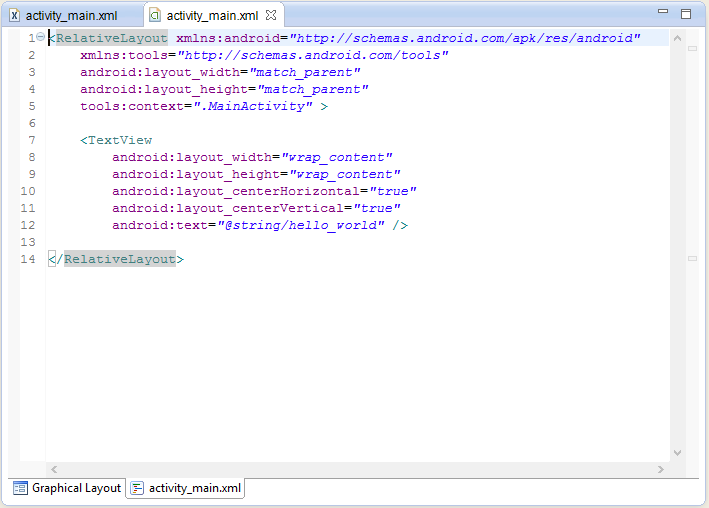
**Bước 5**: Tiếp theo, cửa sổ Config Launch Icon cho phép ta chọn icon đại diện cho ứng dụng. Có thể chọn “Browser” để sử dụng Icon riêng của mình. Trong trường hợp dự án Hello World này chúng tôi dữ nguyên thiết lập mặc định. Chọn Next.

**Bước 6:** Tại bước cuối cùng này sẽ cho phép ta chọn một số kiểu Activity có sẵn, chọn BlankActivity để tạo một Activity đơn giản. Chọn Finish để kết thúc quá trình tạo dự án.

Khi đã thực hiện thành công 06 bước ở trên, tại khung Project Exployer của Eclipse sẽ có một dự án có tên HelloWord, tìm đến thư mục res/layout, đây là thư mục quản lý các tập tin thiết kế giao diện của dự án Android 🡺 chọn tập tin activity\_main.xml. Eclipse cho phép ta thiết kế giao diện bằng 2 cách là Design Graphic theo cơ chế kéo thả (Drag and Drop) và định nghĩa tập tin XML bằng tay. Trong khung làm việc của Eclipse chọn **Graphic Layout** (Hình 2.1.4-3) để vào chế độ thiết kế bằng giao diện và chọn **activity\_main.xml** (Hình 2.1.4-4)để thiết kế bằng tập tin XML.



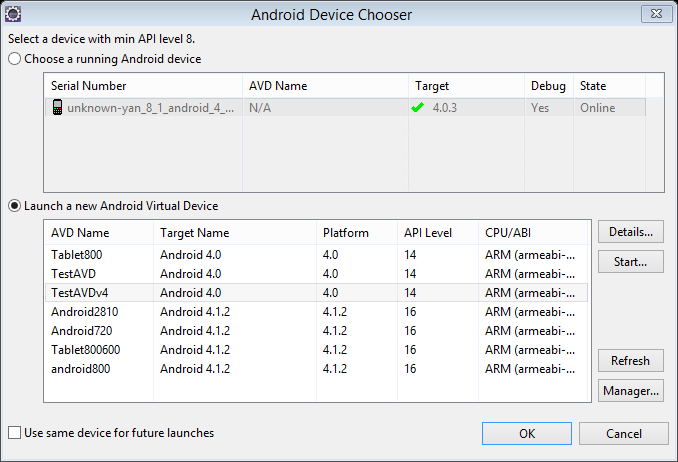
Hình ‑ Xây dựng giao diện sử dụng Graphic Layout

****

Hình ‑ Xây dựng giao diện sử dụng XML

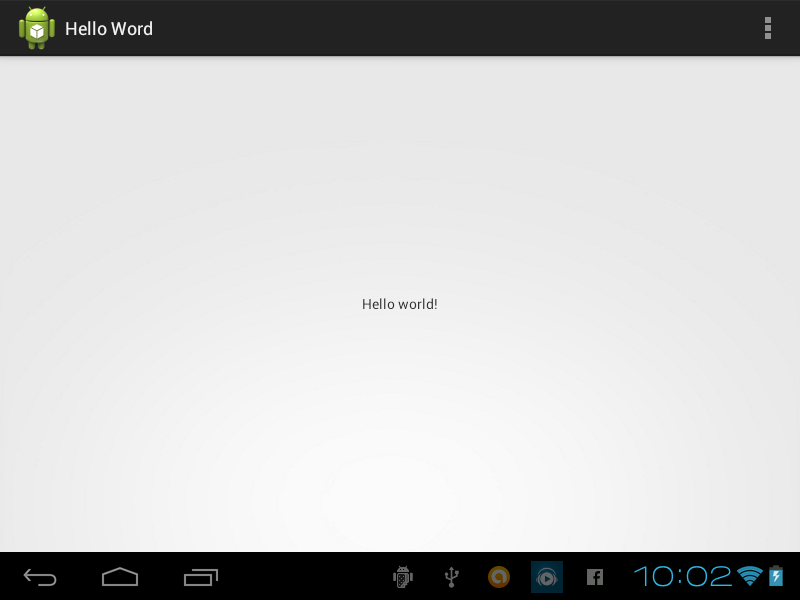
**Bước 7**: Để tiến hành chạy một dự án Android ta chọn tên dự án ở khung Project Exployer 🡺 Nhấp chuột phải 🡺 chọn Run as 🡺 Run Androi Application.

Tiếp theo cửa sổ Android Device Chooser hiện thị danh sách các thiết bị hiện có bao gồm cả thiệt bị thật và ADV. Trong trường hợp đề tài này, ngoài các máy ảo ADV chúng tôi còn sử dụng thiết bị máy tính bảng thương hiệu YAN Tablet chạy phiên bản Android 4.0.3 do đó ở phần **Choose a running Android device** sẽ hiển thị tên của thiết bị thật đang được kết nối với máy tính.



Hình ‑ Cửa sổ chọn thiết bị để chạy dự án Android

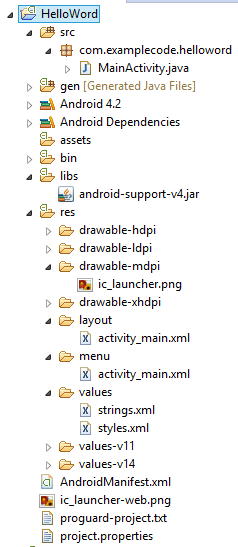
Chọn 01 thiết bị phù hợp và nhấp OK. Trong trường hợp này chúng tôi chọn AVD có tên TestAVDv4 đã được tạo trong phẩn 2.1.3.



Hình ‑ Kết quả ứng dụng HelloWorld

### Phân tích cấu trúc thư mục của một dự án Android

Khi tạo ứng dụng Android HelloWorld được trình bày ở phần trên, ta sẽ thấy cấu trúc thư mục của một dự án Android ở khung Project Exployer của Eclipse như hình 2.1.5-1 .

* src: Chứa những tập tin mã nguồn Java. Trong hình bên chỉ có 01 tập tin **MainActivity.java**, tập tin này là mã nguồn xử lý cho một Activity. Tập tin được lưu tại Package trùng với thuộc tính Project Name khi tạo mới dự án.

Hình ‑ Cấu trúc thư mục dự án Android

* gen: Chưa tập tin **R.java**, một tập tin đặc biệt được tự phát sinh cho phép ta tham khảo đến các tài nguyện của dự án. Tất cả tài nguyên sẽ được tự động biên dịch vào trong lớp này nên ta có thể sử dụng các tài nguyên thông qua lớp này.
* Android 4.x: Thư mục này chưa một tập tin có tên **android.jar**, tập tin này chưa tất cả những lớp thư viện cần thiết cho một ứng dụng Android.
* assets: Thư mục chứa những tập tin bên ngoài được sử dụng trong ứng dụng như tập tin HTML, tập tin Text, cơ sở dữ liệu vân vân.
* bin: Thư mục chưa các tập tin được xây dựng bởi ADT trong tiến trình xây dựng dự án. Thông thường, một tập tin có đuôi .apk (Android Package) sẽ được phát sinh và ta có thể sử dụng tập tin này để chạy ứng dụng Android.
* res: Thư mục này chưa tất cả những tài nguyên được dùng trong ứng dụng. Nó cũng chưa một số thư mục bên trong như **drawable, layout và values**, chúng tôi sẽ trình bày ý nghĩa của những thư mục này ở phần sau.
* AndroidMenufest.xml: Đây là tập tin quan trọng của ứng dụng Android, ta sẽ chỉ định những quyền đặc biệt cho ứng dụng và một vài chức năng khác. Các Activity cũng được khai báo trong tập tin này.

## Các thành phần quản lý giao diện Activities, Fragments, và Intent

Nội dung chính

* + - * Vòng đời của một Activity
      * Sử dụng Fragments để hiệu chỉnh giao diện.
      * Sử dụng Activity như một Dialog.
      * Trình bày khái niệm về Intent.
      * Sử dụng đối tượng Intent để liên kết các Activity.
      * Cách thức sử dụng Intent-filter để kết nối đến những Activities khác nhau.
      * Tạo các thông báo cho người dùng.

Ở phần trước, chúng tôi đã trình bày cách thức để tạo một ứng dụng đơn giản với tập tin tên là **MainActivity.java** cùng với tập tin hiển thị giao diện **activity\_main.xml.** Tập tin MainActivity.java được gọi là một Activity có chức năng hiển thị và xử lý những sự kiện xảy ra trên giao diện được định nghĩa trong tập tin activity\_main.xml. Có thể hiểu Activity là một cửa sổ chứa giao diện của ứng dụng. Thông thường, ứng dụng thường có 01 hoặc nhiều Activity, và mục đích chính của Activity là tương tác với người dùng. Từ lúc Activity xuất hiện đến lúc nó bị ẩn đi, Activity sẽ trải qua một số trạng thái thường được gọi là vòng đời của Activity. Hiểu về vòng đời của Activity là điều cần thiết để có thể lập trình trên Android.

Ngoài Activity, Android 4.0 còn hỗ trợ chức năng phân mảnh giao diện hay được gọi là **Fragment**. Fragment là những Activity thu nhỏ, chúng có thể được gom lại trong một Activity lớn hơn. Trong phần này chúng tôi sẽ trình bày cách thức để kết hợp giữa Fragment và Activity.

Bên cạnh đó, Android 4.0 còn có thêm một khái niệm chỉ có ở hệ điều hành Android đó là **Intent**. Một đối tượng Intent về cơ bản là cách thức để cho phép những Activity khác nhau từ các ứng dụng khác nhau có thể làm việc với nhau một cách liền mạch.

### Một số kiến thức về Activity

Ở phần 2.1 khi tạo ứng dụng HelloWorld, ta đã có một Activity tên là MainActivity.java. Nhấp đôi vào tập tin này ta được các dòng mã như sau

**public** **class** MainActivity **extends** Activity {

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

}

Để một Class Java được hiểu là Activity thì Class đó phải **extend** từ đối tượng Activity. Đối tượng Activity này tải các thành phần giao diện bằng cách sử dụng một tập in XML được định nghĩa trong thư mục res/layout. Trong ví dụ HelloWorld thì MainActivity tải giao diện lên từ tập tin activity\_main.xml bằng phương thức setContentView(int layoutId)

setContentView(R.layout.*activity\_main*);

Ta có thể tạo nhiều Activity khác nhau tương tự như MainActivity. Tất cả các Activity trong ứng dụng đều phải được khai báo trong tập tin **AndroidMainfesh.xml**. Đây là một tập tin quan trọng sẽ được sử dụng nhiều lần trong ứng dụng Android. Ngoài việc được dùng để khai báo các Activity, tập tin này còn được dùng để cấu hình một số quyền truy xuất cho ứng dụng cũng như phiên bản SDK cần thiết để chạy ứng dụng.

<activity

**android:name**=*"com.examplecode.helloword.MainActivity"*

a**ndroid:label**=*"@string/app\_name"* >

<intent-filter>

<action **android:name**=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

Một Activity khai báo trong AndroidMainfesh.xml cần chú ý một số thuộc tính sau

* Android:name: Chỉ định vị trí của Activity
* Android:label: Tên của Activity được hiển thị trên thanh Action Bar của ứng dụng.
* <action android:name>: đặt tên cho Activity thông qua đối tượng Intent, tên này được sử dụng để kết nối Activity với các Activity khác.

Đối tượng trừu tượng **Activity** được khai báo một loạt các phương thức đại diện cho những sự kiện xảy ra trong vòng đời của Activity.

* onCreate(): Được gọi khi Activity được tạo ra lần đầu tiên.
* onStart(): Được gọi khi Activity hiển thị cho người dùng.
* onResume(): Được gọi khi Activity bắt đầu tương tác với người dùng.
* onPause(): Được gọi khi Activity hiện tại được tạm dừng và một Activity khác được bắt đầu chạy.
* onStop(): Được gọi khi không còn được hiển thị cho người dùng.
* onDestroy(): Được gọi trước khi Activity bị hủy.
* onRestart(): Được gọi khi một Activity đã dừng và đang được khởi động lại.

Khi một Activity được khởi chạy, phương thức onStart() và onResume() luôn luôn được gọi đến, bất kể Activity đó đã tồn tại trước đó hay chưa. Khi Activity được tạo lần đầu tiên, phương thức onCreate() sẽ được gọi.

### Hiển thị cửa sổ Dialog sử dụng Activity

Trong khi phát triển ứng dụng Android, có những lúc ta cần hiểu thị một cửa sổ Dialog để yêu cầu người dùng xác nhận, trong trường hợp này ta có thể viết lại phương thức được định nghĩa trong lớp trừu tượng Activity tên là onCreateDialog() để hiển thị một cửa sổ Dialog. Sau đây là hướng dẫn để tạo một Dialog đơn giản.

**Bước 1:** Dùng Eclipse tạo một dự án Android đặt tên là DialogActivity.

**Bước 2:** Mở tập tin res/layout/**main.xml** và thêm vào đoạn mã sau đây.

<Button

android:id=*"@+id/btn\_dialog"*

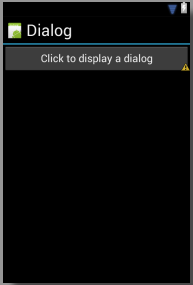
android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text="Click to display a dialog"

android:onClick=*"onClick"* />

Sau đó chọn chuyển chế độ sang **Graphic Layout** ta sẽ được màn hinh như hình 2.2.2-1.



**Bước 3**: Mở tập tin DialogActivity.java trong thư mục src lên và thêm vào phương thức onClick(View v)Đây là một trong những cách để gắn sự kiện cho một thành phần giao diện, chi tiết sẽ được trình bày ở phần tiếp theo.

**public** **void** onClick(View v) {

showDialog(0);

}

**Bước 4**: Viết lại phương thức onCreateDialog (int i).Với nội dung như đoạn mã dưới đây.

Hình ‑ Dialog

@Override

**protected** Dialog onCreateDialog(**int** id) {

**return** **new** AlertDialog.Builder(**this**)

.setIcon(R.drawable.*ic\_launcher*)

.setTitle("This is a dialog with some simple text...")

.setPositiveButton("OK",

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,**int** whichButton)

{

Toast.*makeText*(getBaseContext(),

"OK clicked!", Toast.*LENGTH\_SHORT*).show();

}

}

)

.setNegativeButton("Cancel",

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** whichButton)

{

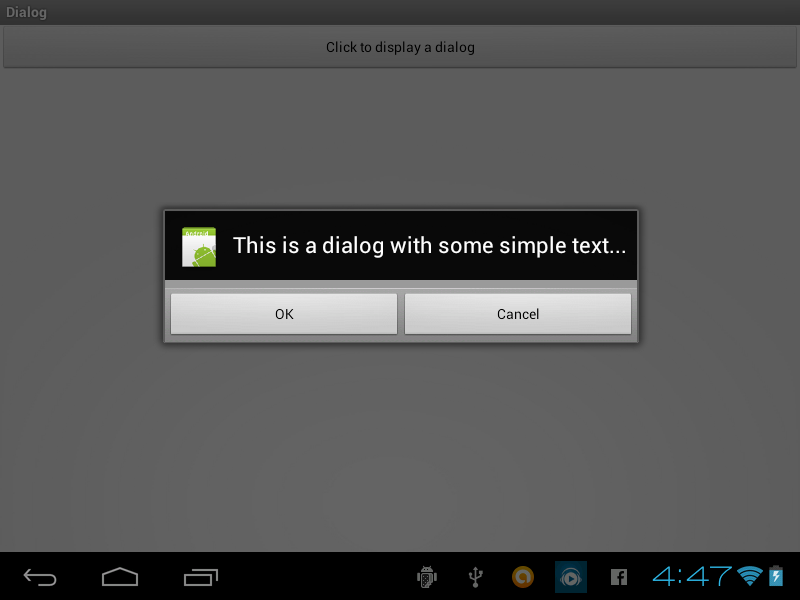
Toast.*makeText*(getBaseContext(),

"Cancel clicked!", Toast.*LENGTH\_SHORT*).show();

}

}

).create(); }

Sau đó ấn tổ hợp phím Ctrl F11 để chạy dự án ta được kết quả như sau khi nhấp vào nút “Click to display dialog”.

Hình ‑ Kết quả hiển thị Dialog đơn giản.

**Giải thích cách thức hoạt động**

Để hiển thị Dialog trước tiên ta cần thực hiện phương thức onCreateDialog() trong lớp Activity. Phương thức này sẽ được gọi thông qua một phương thức khác là showDialog(), phương thức showDialog sẽ được gọi đến khi người dùng nhấp vào nút “Click on to display dialog” trên giao diện.

android:onClick=*"onClick"* // Thiết lập sự kiện khi người dùng nhấp vào nút.

**public** **void** onClick(View v) {

showDialog(0); // Gọi lệnh hiển thị Dialog

}

Phương thức onCreateDialog() được quản lý bởi Activity, khi phương thức showDialog được gọi đến thì phương thức onCreateDialog() cũng được gọi theo. Trong phương thức onCreateDialog() ta sẽ xây dựng một đối tượng Dialog bằng cách sử dụng đối tượng được Android cung cấp : AlertDialog.Builder. Đối tượng này cho phép ta thiết lập những thuộc tính khác nhau như Icon, tiêu đề và các nút của Dialog.

**return** **new** AlertDialog.Builder(**this**)

.setIcon(R.drawable.*ic\_launcher*) // Thiết lập Icon

.setTitle("This is a dialog with some simple text...") // Tiêu đề

.setPositiveButton("OK", // Tạo nút OK

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,**int** whichButton)

{ // Xử lý khi người dùng nhấp nút OK }

} )

.setNegativeButton("Cancel", // Tạo nút Cancel

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog, **int** whichButton)

{ // Xử lý khi người dùng nhấp nút Cancel }

}).create();

### Liên kết các Activity bằng đối tượng Intent

Một ứng dụng Android có thể có một hoặc nhiều Activity, trong trường hợp ứng dụng có nhiều hơn 01 Activity ta thường có nhu cầu di chuyển từ Activity này sang Activity khác. Đối với Android việc di chuyển giữa các Activity được thực hiện thông qua đối tượng được gọi là Intent.

Intent là một vấn đề rất quan trọng trong lập trình Android nhưng định nghĩa của nó thì khá trừu tượng nên chúng tôi sẽ chỉ nói về phương pháp lập trình để có thể liên kết các Activity lại với nhau thông qua ví dụ dưới đây.

Ví dụ này sẽ hướng dẫn cách thêm một Activity khác vào dự án và di chuyển qua lại giữa 02 Activity.

**Bước 1:** Dùng Eclipse tạo một dự án mới đặt tên là UsingIntent.

**Bước 2:** Nhấp chuột phải vào gói được đặt trong thư mục src 🡺 chọn New 🡺 chọn Class. Bước này được thực để tạo ra một Activity mới bằng cách extends lớp trừu tượng Activity. Cửa sổ New Class xuất hiện, ta đặt tên cho Activity mới này là SecondActivity.

**Bước 3:** Mở tập tin AndroidMainfesh.xml và khai báo một Activity mới theo đoạn mã như sau.

<activity

android:label=*"Second Activity"*

android:name=*".SecondActivity"* >

<intent-filter >

<action android:name=*"net.learn2develop.SecondActivity"* />

<category android:name=*"android.intent.category.DEFAULT"* />

</intent-filter>

</activity>

**Bước 4:** Copy tập tin res/layout/main.xml và Paste lại vào thư mục res/layout, sau đó đổi tên lại thành secondactivity.xml.

**Bước 5:** Thêm vào nội dung của tập tin secondactivity.xml như sau.

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"* >

**<TextView**

**android:layout\_width=*"fill\_parent"***

**android:layout\_height=*"wrap\_content"***

**android:text=*"This is the Second Activity!"* />**

</LinearLayout>

**Bước 6:** Mở lớp SecondActivity.java đã tạo ở bước 1 và sửa nội dung lại.

**public** **class** SecondActivity **extends** Activity{

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

**setContentView(R.layout.*secondactivity*);**

}

}

**Bước 7:** Tại tập tin main.xml ta thêm vào giao diện một nút và thiết lập cho nó sự kiện onClick.

**<Button**

**android:layout\_width=*"fill\_parent"***

**android:layout\_height=*"wrap\_content"***

**android:text=*"Display second activity"***

**android:onClick=*"onClick"*/>**

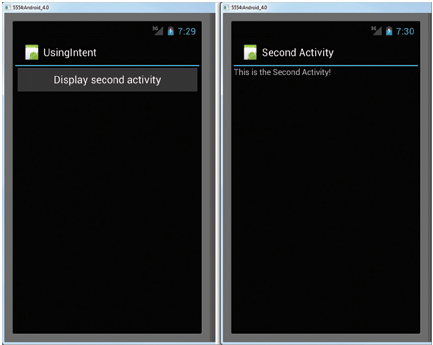
**Bước 8**: Thêm sự kiện onClick vào tập tin UsingIntentActivity.java

**public** **void** onClick(View view) {

startActivity(**new** Intent("net.learn2develop.SecondActivity"));

}

**Bước 9:** Chạy ứng dụng, khi Activity đầu tiên được tải lên ta nhấp chọn nút “Display second activity” và Activity thứ hai lúc này sẽ được hiển thị.



Hình ‑ Liên kết 02 Activity dùng Intent

**Giải thích cách thức hoạt động**

Như đã biết, một Activity được tạo bởi một thành phân giao diện (main.xml) và một lớp xử lý (UsingIntentActivity.java). Dó đó, nếu muốn thêm một Activity khác vào dự án, ta cần phải tạo cả hai thành phần này. Trong tập tin AndroidManifest.xml ta khai báo một Activity mới dùng thẻ <activity>

<activity

android:label=*"Second Activity"*

android:name=*".SecondActivity"* >

**<intent-filter >**

**<action android:name=*"net.learn2develop.SecondActivity"* />**

**<category android:name=*"android.intent.category.DEFAULT"* />**

**</intent-filter>**

</activity>

Ở đây, khi thêm một Activity mới cho ứng dụng ta cần chú ý đến cách sử dụng thẻ <intent-filter>.

* Thuộc tính <action android:name> cho Activity mới được dùng để một Activity khác thông qua tên này sẽ gọi được đến Activity.
* Thuộc tính <category> được thiết lập là android.intent.categoru.DEFAULT. Ta cần thêm thuộc tính này cho intent-filter để Activity này có thể được khởi chạy từ một Activity khác bằng cách sử dụng phương thức startActivity().

Khi người dùng nhấp vào nút, ta sử dụng phương thức startActivity() để hiển thị giao diện của SecondActiviy bằng cách tạo một thể hiện của lớp Intent và truyền cho nó tên của intent-filter được khai báo trong tập tin AndroidManifesh.xml (trong trường hợp này là net.learn2develop.SecondActivity).

**public** **void** onClick(View view) {

startActivity(**new** Intent("net.learn2develop.SecondActivity"));

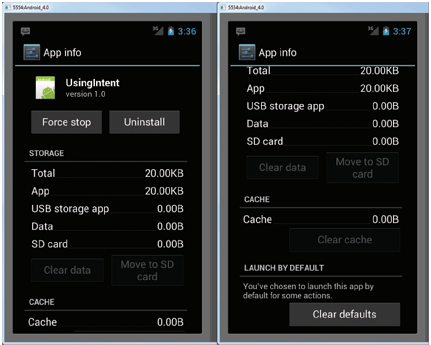
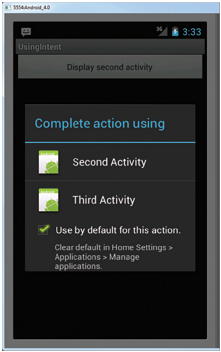
}

Các đối tượng Activity trong Android có thể được gọi bởi bất ký ứng dụng nào đang chạy trong cùng một thiết bị. Ví du, ta có thể tạo một dự án Android mới và hiển thị Second-Activity bằng cách khai bào một intent-filter có name giống với SecondActivity. Đây là một trọng những khái niệm cơ bản của Android cho phép các ứng dung có thể gọi đến nhau một cách dễ dàng.

**Xử lý đụng độ khi sử dụng Intent**

Trong trường hợp có nhiều Activity được khai bảo cùng 01 tên trong intent-filter thì hệ điều hành Android sẽ hiển thị danh sách các Activity đó để người dùng lựa chọn (Hình 2.2.3-2). Nếu ta chọn “Use by default for this action” thì khi được gọi vào lần tiếp theo ứng dụng sẽ tự động chuyển đến Activity được chọn làm mặc định vào lần gọi trước.

Để xóa lựa chọn mặc định này ta sử dụng ứng dụng **Settings** của Android và chọn 🡺 Apps 🡺 Manager application 🡺 chọn tên ứng dụng cần xóa. Khi màn hình hiển thị chi thông tin chi tiết của ứng dụng, cuốn xuống cuối và nhấp vào nút “Clear defaults”.

. 

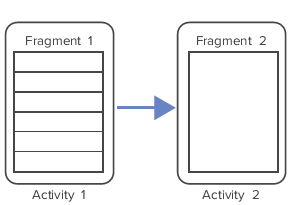
Hình ‑ Clear Default

Hình ‑

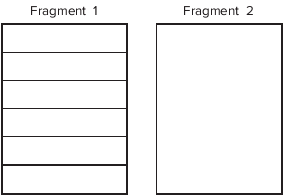
### Fragments

Ở những phần trước ta đã biết được Activity là gì và cách sử dụng chúng. Đối với các thiết bị với màn hình nhỏ như điện thoại hay máy tính bảng 7 inch, một Activity thường sẽ sẽ đầy màn hình, hiển thị nhiều thành phần giao diện của ứng dụng. Về cơ bản, Activity là một vật chứa các thành phần hiển thị. Tuy nhiên, khi một Activity có quá nhiều thành phần cần quản lý trong khi kích thước của màn hình lại quá nhỏ hoặc ngược lại, khi kích thước màn hình quá lớn và 1 Activity không thể làm đầy nó thì cách hiệu quả nhất là sử dụng nhưng Activity thu nhỏ (mini-activities), mỗi Activity thu nhỏ này sẽ có phần hiển thị giao diện và xử lý độc lập, không phụ thuộc lần nhau. Trong thời gian runtime, một Activity có thể chứa một hoặc nhiều mini-activities phù thuộc vào kích thước của màn hình. Từ phiên bản Android 3.0 trở đi, những mini-activities này được biết đến với tên Fragment.

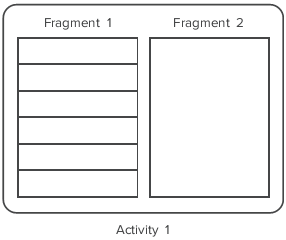
Có thể hiểu, Fragment là một dạng khác của Activity, ta cũng tạo những Fragment để chứa các thành phần giao diện. Fragment luôn luôn được nhúng vào trong Activity, ví dụ ở hình dưới có 02 Fragment. Fragment 1 chứa một ListView, Fragment 2 chứa một số thành phần khác như TextView và ImageView (Hình 2.2.4-1). Fragment 1 được chứa trong Activity 1, Fragment 2 được chứa trong Activity 2. Khi người dùng chọn một mục trong danh dánh ở Fragment 1 thì Activity 2 được khởi chạy.



Hình ‑ Activity 1 và 2



Hình ‑ Fragments 1 và 2

Nếu ứng dụng được chạy trên màn hình máy tính bảng nằm ngang thì cả 2 Fragment sẽ được nhúng vào trong 1 Activity như hình 2.2.4-3.

Hình ‑ 2 Fragment trong 1 Activity

Có 02 cách để thêm Fragment vào trong một Activity :

* Chỉ định sử dụng Fragment bằng XML.
* Thêm Fragment trong thời gian runtime sử dụng đối tượng FragmentManager.

**Chỉ định sử dụng Fragment bằng XML**

**Bước 1**: Tạo dự án mới đặt tên là Fragments

**Bước 2**: Trong thư mục res/layout tạo 02 tập tin mới đặt tên là fragment1.xml và fragment2.xml.

Nội dung của fragment1.xml

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:background=*"#00FF00"*

>

<TextView

android:id=*"@+id/lblFragment1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"This is fragment #1"*

android:textColor=*"#000000"*

android:textSize=*"25sp"* />

</LinearLayout>

Nội dung của fragment2.xml.

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:background=*"#FFFE00"*

android:orientation=*"vertical"* >

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"This is fragment #2"*

android:textColor=*"#000000"*

android:textSize=*"25sp"* />

</LinearLayout>

**Bước 3**: Thêm đoạn mã dưới đây vào tập tin main.xml trong thư mục res/layout.

<fragment

**android:name=*"net.learn2develop.Fragments.Fragment1"***

**android:id=*"@+id/fragment1"***

android:layout\_weight=*"1"*

android:layout\_width=*"0px"*

android:layout\_height=*"match\_parent"* />

<fragment

**android:name=*"net.learn2develop.Fragments.Fragment2"***

**android:id=*"@+id/fragment2"***

android:layout\_weight=*"1"*

android:layout\_width=*"0px"*

android:layout\_height=*"match\_parent"* />

**Bước 4**: Trong thư mục src/net.learn2develop.Fragments thêm vào lớp có tên là Fragment1.java và Fragment2.java có nội dung như sau

Nội dung Fragment1.java

**public** **class** Fragment1 **extends Fragment** {

@Override

**public** View onCreateView(**LayoutInflater inflater**,

ViewGroup container, Bundle savedInstanceState) {

**return** **inflater.inflate(**

**R.layout.*fragment1*, container, false);**

}

}

Nội dụng Fragment2.java

**public** **class** Fragment2 **extends** Fragment {

@Override

**public** View onCreateView(LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

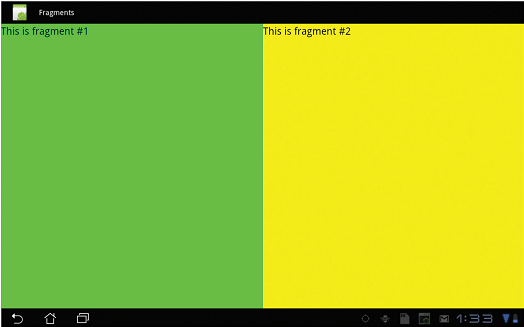
**return** inflater.inflate(

R.layout.*fragment2*, container, **false**);

}

}

**Bước 5**: Chạy ứng dụng, và ta thấy rằng khi ứng dụng khởi động thì FragmentsActivy sẽ được gọi và có hai Fragment được nhúng trong FragmentsActivity.



Hình ‑ Nhúng Fragment1 và Fragment2 vào 01 Activity

**Giải thích cách thức hoạt động**

Một Fragment có cách thực hiện và những ứng xử rất giống với một Activity. Fragment cũng được tạo lên từ 02 bộ phận : 1 lớp Java để xử lý sự kiện và một tập tin XML để tải giao diện người dùng.

Tập tin XML có thể chứa tất cả những thành phần giao diện thông thường mà ta muốn như TextView, EditText, Button… Lớp Java của Fragment cần phải được kế thừa bởi lớp Fragment trừu tượng của Android.

**public** **class** Fragment1 **extends Fragment** {}

Để hiển thị giao diện cho Fragment, ta viết lại phương thức onCreateView(). Phương thức này trả về một đối tượng View.

**public** View onCreateView(**LayoutInflater inflater**,

ViewGroup container, Bundle savedInstanceState) {

**return** **inflater.inflate(**

**R.layout.*fragment1*, container, false);**

}

Ở đây ta sử dụng đối tượng LayoutInflater để lấy nội dung của tập tin giao diện XML bằng cách dùng đối tượng quản lý tài nguyên R. Trong ví dụ này thì Fragment1 sẽ hiển thị giao diện được định nghĩa trong tập tin fragment1.xml.

Để thêm một Fragment vào Activity ta sử dụng thẻ <fragment> với thuộc tính name tham khảo đến vị trí của tập tin Fragment1.java.

<fragment

**android:name=*"net.learn2develop.Fragments.Fragment1"/>***

**Thêm Fragment sử dụng FragmentManager**

Khi ứng dụng các Fragment, ta có thể chia nhỏ giao diện thành nhiều phần với các cấu hình khác nhau. Điểm đặc biệt nhất của Fragment là ta có thể thêm chúng vào các Activity trong thời gian Runtime của ứng dụng. Điều này giúp ta có thể thay đổi giao diện tùy ý để phù hợp với ngữ cảnh của ứng dụng.

**Bước 1:** Mở tập tin **main.xml** trong thư muc res/layout và xóa đoạn mã khai báo thẻ <fragment> trước đó.

**Bước 2:** Trong tập tin FragmentActivity.java thêm vào đoạn mã sau trong phương thức onCreate().

**FragmentManager** fragmentManager = getFragmentManager();

**FragmentTransaction** fragmentTransaction = fragmentManager.beginTransaction();

WindowManager wm = getWindowManager();

Display d = wm.getDefaultDisplay();

**if** (**d.getWidth() > d.getHeight()**) {

**Fragment1 fragment1 = new Fragment1();**

**fragmentTransaction.replace(android.R.id.*content*, fragment1);**

} **else** {

**Fragment2 fragment2 = new Fragment2();**

**fragmentTransaction.replace(android.R.id.*content*, fragment2);**

}

fragmentTransaction.commit();

**Bước 3**: Chạy ứng dụng, Khi chạy ứng dụng nếu để máy tính bảng nằm ngang thì ứng dụng sẽ hiển thị giao diện của Fragment1 và ngược lại, nếu đề thiết bị theo chiều đứng thì giao diện được hiển thị là giao diện của Fragment2.

**Giải thích cách thức hoạt động**

Để thêm Fragment cho một Activity ta sử dụng đối tượng FragmentManager và lấy một thể hiện của đối tượng này thông qua phương thức getFragmentManager().

**FragmentManager** fragmentManager = getFragmentManager();

Ta cũng cần sử dụng một đối tượng nữa là FragmentTransaction để thực hiện những giao tác trong Activity như là thêm, xóa hoặc thay thế các Fragment.

**FragmentTransaction** fragmentTransaction = fragmentManager.beginTransaction();

Trong ví dụ này, ta sử dung đối tượng WindownManager để xác định thiết bị đang được đặt đặt nằm ngang hay đặt đứng. Sau đó ta tạo Fragment phù hợp và gọi làm replace() của đối tượng FragmentTransaction để thêm Fragment vào cho Activity.

**Fragment1 fragment1 = new Fragment1();**

**fragmentTransaction.replace(android.R.id.*content*, fragment1);**

Trong hàm replace ta truyền vào ID của thành phần giao diện sẽ được nhúng Fragment vào. Sử dụng phương thức replace() thực chất là gọi phương thức remove() để xóa Fragment đang có trong thành phần giao diện sau đó gọi phương thức add() để thêm Fragment. Cuối cùng, để đảm bảo những thay đổi có hiệu lực ta gọi đến phương thức commit().

fragmentTransaction.commit();

## Các thành phần giao diện người dùng của Android

Nội dung chính

* Các loại ViewGroups khác nhau có thể được dùng để bố trí thành phần giao diện (View).
* Cách thức để tạo giao diện người dùng bằng đối tượng Java.
* Cách thức đến lắng nghe và xử lý các sự kiện khi người dùng tương tác với giao diện.

Trong phần 2.2, chúng tôi đã nghiên cứu về các Activiy và vòng đời của chúng. Ta biết rằng một Activity được dùng để tương tác với người dùng. Tuy nhiên, một Activity không thể tự tạo nên giao diện cho người dùng được mà nó sử dụng các đối tượng khác để hiển thị, những đối tượng này được gọi là Views và ViewGroúp. Trong phần này, ta sẽ tìm hiểu chi tiết hơn về cách tạo giao diện người dùng trên Android cũng như cách thức mà người dùng tương tác với giao diện.

### Một số kiến thức về các thành phần giao diện: View, ViewGroup và Layout

Như ta đã biết, đơn vị căn bản của một ứng dụng Android là Activity. Activity có thể chứa những thành phần như nút ấn, ký tự, hộp kiểm v..v.. Thông thường ta định nghĩa giao diện người dùng bằng cách dùng tập tin XML đặt trong thư mục res/layout sau đó trong lớp Activity.java ta dùng hàm **setContentView()** để gắn tập tin XML cho Activity.

Trong quá trình biên dịch, mỗi thành phần được định nghĩa trong tập tin XML sẽ được biên dịch vào những lớp giao diện Android tương ứng với các thuộc tính được đại diện thông qua các phương thức. Sau đây ta cùng tìm hiểu một số khái niệm về các thành phần giao diện trong Android để có cái nhìn tổng quan và xây dựng phương hướng cho việc phát triển giao diện ứng dụng trên Android.

**Views và ViewGroups**

View được hiểu là một đơn vị giao diện được hiển thị trên màn hình. Ví dụ View là nút ấn, ký tự hay các hộp kiểm.

Một hoặc nhiều View có thẻ được gom nhóm lại với nhau trong một View lớn hơn gọi là ViewGroup. ViewGroup là một loại View đặc biệt cung cập nột bố cục nhất định cho phép ta có thể xắp xếp cách hiển thị và thứ tự các View.

Android hỗ trợ cho ta một số loại ViewGroup dưới đây :

* LinearLayout
* AbsoluteLayout
* TableLayout
* RelativeLayout
* FrameLayout
* ScrollView.

Trong phạm vi đề tài này chúng tôi sẽ trình bày về 03 loại ViewGroup thường được sử dụng nhất đó là LinearLayout, RelativeLayout và ScrollView.

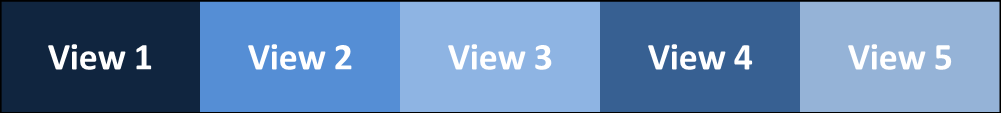
**LinearLayout**

LinearLayout xắp xếp các View thành một hàng đơn hoặc là một cột đơn, các View con có thể được sắp xếp theo chiều thẳng đứng (vertically) hoặc nằm ngang (horizontally).

Để quy định cho các View bên trong được xắp xếp theo chiều thẳng đứng hay nằm ngang ta dùng thuộc tính android:orientation để quy định

LinearLayout với các View bên trong nằm ngang (Hình 2.3.1-1)

<LinearLayout android:orientation=*"horizontal"*> .... </LinearLayout>



Hình ‑ LinearLayout ngang

LinearLayout với các View thẳng đứng (Hình 2.3.1-2).

<LinearLayout android:orientation=*"vertical"*> .... </LinearLayout>

Ngoài ra ta cũng có thể kết hợp nhiều LinearLayout với chiều khác nhau như lồng một LinearLayout nằm ngang trong một LinearLayout thẳng đứng (Hình 2.3.1-3).

<LinearLayout android:orientation=*"vertical"*>

....

<LinearLayout android:orientation=*"horizontal"*> .... </LinearLayout>

....

</LinearLayout>

|  |  |
| --- | --- |
| Picture2  Hình ‑ LinearLayout thẳng đứng | Picture3  Hình ‑ LinearLayout hỗn hợp |

Đoạn mã sau đây minh họa cho việc sắp xếp các View trong LinearLayout.

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android "*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

**android:orientation=*"vertical"*** >

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:padding=*"5dip"*

android:text=*"Email:"* />

<EditText

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginBottom=*"10dip"* />

<Button

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Login"* />

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginTop=*"25dip"*

android:background=*"#2a2a2a"*

**android:orientation=*"horizontal"***>

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"1"*

android:gravity=*"center"*

android:padding=*"15dip"*

android:text=*"Home"* />

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_weight=*"1"*

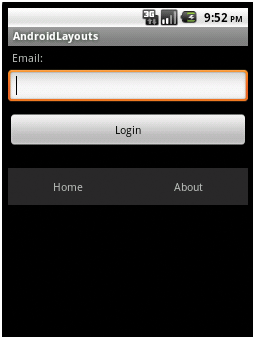
android:gravity=*"center"*

android:padding=*"15dip"*

android:text=*"About"* />

</LinearLayout>

</LinearLayout>

Trong đoạn mã trên, ta sử dụng một LinearLayout lớn được chỉ định thuộc tính orientation= “vertical”. LinearLayout này chứa các loại View khác nhau như: TextView, EditTextView và Button. Đặc biệt, ta lồng vào trong đó một LinearLayout khác (vùng mã nguồn được tô màu) được chỉ định orientation= “horizontal” và chứa 02 TextView. Khi đó LinearLayout ở trong sẽ được hiểu như là một View con của LinearLayout nằm ngoài. Ta có kết quả như hình 2.3.1-4.

Hình ‑ ví dụ LinearLayout

Sau đây là bảng liệt kê một số thuộc tinh chung của View và ViewGroup

Bảng ‑ Một số thuộc tính chung của View và ViewGroup

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| **Layout\_width** | Chỉ định độ dài của View hoặc ViewGroup |
| **Layout\_height** | Chỉ định độ cao của View hoặc ViewGroup |
| **Layout\_marginTop** | Chỉ định khoảng trống được thêm phía trên. |
| **Layout\_marginBottom** | Chỉ định khoảng trống được thêm phía dưới. |
| **Layout\_marginLeft** | Chỉ định khoảng trống được thêm bên trái |
| **Layout\_marginRight** | Chỉ định khoảng trống được thêm bên phải |
| **Layout\_gravity** | Chỉ định vị trí của các View bên trong |
| **Layout\_weight** | Chỉ định không gian mà View chiếm. |
| **Layout\_x** | Chỉ định tọa độ x của góc trên bên trái |
| **Layout\_y** | Chỉ định tọa độ y của gọc trên bên trái |

**RelativeLayout**

RelativeLayout là một ViewGroup hiển thị các View thành phần dựa trên mối liên hệ về vị trí. Vị trí của mỗi View có thể được chỉ định dựa trên mối liên hệ với các View lân cận với nó (ví dụ nằm bên trái hay nằm dưới một View khác) hoặc dựa trên mối liên hệ với View chứa nó ( ví dụ: nằm phía dưới bên trái của View chứa nó).

RelativeLayout là một công cụ rất hữu dụng cho việc thiết kế giao diện bởi vì nó có thể loài bỏ việc đặt lồng các ViewGroup với nhau và giữ cho cấu trúc Layout được cố định, việc này sẽ giúp tăng hiệu xuất xử lý khi chạy ứng dụng. Ta có thể thay thể nhiều LinearLayout được lồng với nhau bằng duy nhất một RelativeLayout.

* **Đặt vị trí cho các View**

RelativeLayout chỉ định vị trí cho các View thành phần dựa trên liên hệ của nó với View chứa nó hoặc với các View xung quanh (được chỉ định bằng ID).

Do đó ta có thể sắp xếp các thành phần theo biên của View chứa, hoặc sắp xếp một View nằm dưới một View khác, nằm ở giữa màn hình hoặc lệch qua trái v.v.. Mặc định tất cả các View thành phần sẽ nằm ở góc trên bên trái và ta phải định nghĩa vị trí cho từng View bằng cách dùng các thuộc tính bố cục khác nhau được định nghĩa bởi đối tượng RelativeLayout.LayoutParams.

Một số loại thuộc tính bố cục có sẵn trong RelativeLayout :

android:layout\_alignParentTop

Nếu là “true” thì góc trên của View sẽ trùng với góc trên của View chứa.

android:layout\_centerVertical

Nếu là “true” thì tâm của View sẽ trùng với tâm của View chứa.

android:layout\_below

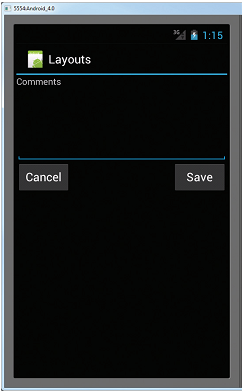
Vị trí cạnh trên của View sẽ nằm dưới View được chỉ định

android:layout\_toRightOf

Vị trí cạnh trái của View sẽ nằm bên phải của View được chỉ định

Đoạn mã minh họa sử dụng RelativeLayout

<RelativeLayout xmlns:android=*"”http://schemas.android.com/apk/res/android”"*

 android:id=*"”@+id/RLayout”"*

Hình ‑ RelativeLayout

android:layout\_width=*"”fill\_parent”"*

android:layout\_height=*"”fill\_parent”"* >

<TextView

android:id=*"”@+id/lblComments”"*

android:layout\_width=*"”wrap\_content”"*

android:layout\_height=*"”wrap\_content”"*

android:layout\_alignParentLeft=*"”true”"*

android:layout\_alignParentTop=*"”true”"*

android:text=*"”Comments”"* />

<EditText

android:id=*"”@+id/txtComments”*

android:layout\_alignLeft=*"”@+id/lblComments”"*

android:layout\_below=*"”@+id/lblComments”"*

android:layout\_centerHorizontal=*"”true”"*

android:textSize=*"”18sp”"* />

<Button

android:id=*"”@+id/btnSave”"*

android:layout\_alignRight=*"”@+id/txtComments”"*

android:layout\_below=*"”@+id/txtComments”"*

android:text=*"”Save”"* />

<Button

android:id=*"”@+id/btnCancel”*

android:layout\_height=*"”wrap\_content”"*

android:layout\_alignLeft=*"”@+id/txtComments”"*

android:layout\_below=*"”@+id/txtComments”"*

android:text=*"”Cancel”"* />

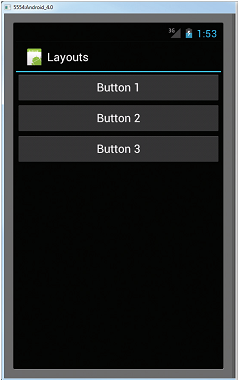
</RelativeLayout>

**ScrollView**

ScrollView là một loại ViewGroup đặc biệt cho phép người dùng có thể cuộn qua lại thông qua một danh sách các View. Điều này giúp cho ta có nhiều không gian hơn là giới hạn màn hình thật sự của thiết bị. ScroollView chỉ có thể chứa duy nhất một phần tử View hoặc ViewGroup, thông thường sẽ chứa một LinearLayout.

Đoạn mã dưới đây giới thiệu cách thức sử dụng ScrollView

<ScrollView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

Hình ‑ ScrollView 1

android:layout\_height=*"fill\_parent"* >

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"* />

<Button

android:id=*"@+id/button2"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"* />

<Button

android:id=*"@+id/button3"*

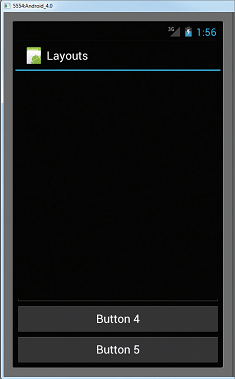
android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"* />

<EditText

android:id=*"@+id/txt"*

 android:layout\_width=*"fill\_parent"*

Hình ‑ ScrollView 2

**android:layout\_height=*"600dp"*** />

<Button

android:id=*"@+id/button4"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"* />

<Button

android:id=*"@+id/button5"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Button"* />

</LinearLayout>

</ScrollView>

Khi chạy đoạn mã trên, màn hình sẽ hiển thị như hình 2.3.1-6, vì ta quy định chiều cao của EditText là 600dp nên tổng độ dài của các thành phần bên trong lơn hơn kích thước màn hình dẫn đến không nhìn thấy 02 nút ở dưới. Sau đó khi ta kéo màn hình lên sẽ nhìn thấy xuất hiện 02 nút này.

### Tạo giao diện người dùng bằng phương pháp lập trình hướng đối tượng

Từ đầu tài liệu này đến nay, tất cả những giao diện được trình bày đều được tạo nên bằng phương pháp sử dụng tập tin XML. Như đã đề cập ở trên, bên cạnh việc sử dụng tập tin XML ta còn có thể tạo giao diện bằng cách dùng mã lập trình. Phương pháp nhà thật sự hữu ích khi ta cần phát sinh động các giao diện trong thời gian Runtime của ứng dụng. Ví dụ, trong ứng dụng đặt món ăn của đề tài này, ứng dụng sẽ hiển thị danh sách các món ăn và khi người dùng chọn món ăn muốn đặt thì ứng dụng sẽ phải phát sinh nội dung theo món ăn được chọn. Trong trường hợp này, ta cần nội dung của giao diện được phát sinh tự động tùy theo lựa chọn của người dùng. Các bước sau đây sẽ trình bày cách thức lập trình cần thiết để xây dựng động giao diện cho Activity.

**Bước 1:** Tạo một dự án Android đặt tên là UICode

**Bước 2:** Trong tập tin UICodeActivity.java ta thêm vào trong phương thức onCreate() đoạn mã sau.



Hình ‑ UICode

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

// setContentView(R.layout.main);

LayoutParams params = **new** LinearLayout.LayoutParams(LayoutParams.*FILL\_PARENT*, LayoutParams.*WRAP\_CONTENT*);

// ---Tạo một LinearLayout---

LinearLayout layout = **new** LinearLayout(**this**);

layout.setOrientation(LinearLayout.*VERTICAL*);

// ---Tạo TextView---

TextView tv = **new** TextView(**this**);

tv.setText("This is a TextView");

tv.setLayoutParams(params);

// ---Tạo Button---

Button btn = **new** Button(**this**);

btn.setText("This is a Button");

btn.setLayoutParams(params);

// ---Gắn TextView vào Layout---

layout.addView(tv);

// ---Gắn Button vào Layout---

layout.addView(btn);

// ---Tạo đối tượng LayoutParam cho Layout---

LinearLayout.LayoutParams layoutParam = **new** LinearLayout.LayoutParams(LayoutParams.*FILL\_PARENT*,LayoutParams.*WRAP\_CONTENT*);

// ---Gắn layout vào giao diện---

**this**.addContentView(layout, layoutParam);

}

**Bước 3:** Chạy ứng dụng ta sẽ thấy giao diện hiển thị như hình 2.3.2-1

**Giải thích cơ chế hoạt động**

Trong ví dụ này, trước tiên ta cần đóng đoạn mã gọi đến phương thức setContentView() để Activity không tải giao diện từ tập tin main.xml. Sau đó ta tạo đối tượng LayoutParam để quy định các thông số về kích thước cho các thành phần bên trong.

LayoutParams params = **new** LinearLayout.LayoutParams( LayoutParams.*FILL\_PARENT*, LayoutParams.*WRAP\_CONTENT*);

Đoạn mã trên tạo ra một đối tượng LayoutParams có chiều dài *FILL\_PARENT* nghĩa là độ dài bằng với độ dài của View chứa và chiều cao *WRAP\_CONTENT*nghĩa là chiều cao bằng với chiều cao của các View thành phần. Đối tượng LayoutParams này có thể được dùng chung cho nhiều View khác nhau.

Sau đó ta tạo một đối tượng LayoutLinear để làm ViewGroup chứa tất cả những View thành phần sẽ được tạo ra sau đó.

LinearLayout layout = **new** LinearLayout(**this**);

layout.setOrientation(LinearLayout.*VERTICAL*);

Khi xây dựng giao diện bằng phương pháp này, một điều cần nhớ là hầu hết tất cả các thuộc tính của View được tạo bằng XML ta đều có thể thiết lập thông qua phương thức Setter tương ứng với thuộc tính đó. Trong đoạn mã trên ta ta thiết lập chiều của layout là thẳng đứng tương dùng **setOrientation(LinearLayout.VERTICAL)** tương ứng với thuộc tính **android:orientation = “vertical”** trong tập tin XML.

Tương tự, ta cũng tạo các View thành phần là TextView và Button.

// ---Tạo TextView---

TextView tv = **new** TextView(**this**);

tv.setText("This is a TextView"); // thiết lập chuỗi hiển thị

tv.setLayoutParams(params); // gắn LayoutParams cho View

// ---Tạo Button---

Button btn = **new** Button(**this**);

btn.setText("This is a Button");

btn.setLayoutParams(params);

Sau đó ta thêm các View thành phần vào trong layout

// ---Gắn TextView vào Layout---

layout.addView(tv);

// ---Gắn Button vào Layout---

layout.addView(btn);

Phương thức addView(View view) được dùng để thêm một View thành phần, nếu View thành phần chưa được quy định LayoutPrarams thì mặc định sẽ lấy LayoutParams của ViewGroup để thiết lập cho View.

Tiếp theo ta tạo LayoutParams cho layout. Cuối cùng ta gắn layout lên giao diện của ứng dụng.

**this**.addContentView(layout, layoutParam);

Phương thức addContentView() được dùng để thêm một nội dung giao diện cho Activity. Nội dung giao diện này sẽ xuất hiện sau những View đã có của Activity. Chú ý là những View đã tồn tại trong Activity sẽ không bị xóa khi dùng phương thức này.

### Lắng nghe và xử lý sự kiện tương tác giao diện

Tương tác giữa người dùng và giao diện được chia làm 02 mức độ: Tương tác mức Activity và tương tác mức View. Ở mức tương tác Activity, lớp trừu Activity cung cấp một số phương thức mà ta có thể viết lại để xử lý những tương tác. Sau đây là một số phương thức thông dụng.

Bảng ‑ Các sự kiện mức Activity

|  |  |
| --- | --- |
| **Tên phương thức** | **Chú thích** |
| **onKeyDown** | Được gọi khi người dùng ấn một nút trên thiết bị |
| **onKeyUp** | Được gọi khi người dùng thả một nút trên thiết bị |
| **onMenuItemSelected** | Đươc gọi khi một mục trong khung Menu được chọn |
| **onMenuOpened** | Được gọi khi Menu được mở lên. |

Trong phạm vi đề tài này chúng tôi không đề cập chi tiết về lắng nghe và xử lý tương tác ở mức Activity mà tập trung vào mức tương tác thứ 2 : tương tác mức View. Trước khi nói về cách thức làm thế nào để xử lý sự kiện trong lập trình chúng tôi sẽ trình bày về các loại sự kiện trong Android.

Có nhiều cách để nhận được sự kiện từ những tương tác của người dùng. Một cách tiếp cận thông dụng để bắt các sự kiện này là thông qua những đối tượng View có khả năng tương tác với người dùng.Trong lớp trừu tượng View có chứa một tập hợp những Interface lồng nhau cùng với các hàm gọi lại (callback) mà giúp ta có thể dễ dàng lắng nghe được các sự kiện xảy ra trên giao diện. Những Interface này được gọi chung là Event Listeners.

**Bắt sự kiện sử dụng Event Listeners**

Một Event Listener là một Interface trong lớp trừu tượng View, trong Interface này có chứa các hàm callback, những hàm callback được gọi bởi Android khi người dùng tương tác lên một View đã được đăng ký lắng nghe sự kiện. Sau đây là một số hàm callback được chứa trong các Event Listener.

* onClick(): được chứa trong Interface View.OnClickListener. Phương thức này được gọi khi người dùng chạm vào View hoặc nhấp vào các phím chức năng như Enter, qua trái-phải- lên xuống.
* onLongClick (): được chứa trong Interface View.OnLongClickLinstener. Được dùng tương tự như onClick() nhưng trong trường hợp người dùng chạm và giữ trong thời gian dài hơn.
* onFocusChange(): được chứa trong Interface View.OnFocusChangeListener. Gọi đến khi người dùng chuyển đi hoặc chuyển đến một View.
* onTouchListener(): được chứa trong Interface View.OnTouchListener. Được gọi đến khi người dùng thực hiện một thao tác nào đó trên màn hình như chạm, giữ, thả hoặc bất cứ tác động nào trên màn hình.
* onCreateContextMenu(): nằm trong Interface View.OnCreateContextMenuListener. Được gọi đến khi Context Menu được tạo nên (được sử dụng với onLongClick).

Để định nghĩa những phương thức này và xử lý các sự kiện, ta có thể cho Activity hiện thực Interface hoặc định nghĩa một lớp vô danh (anonymous class). Sau đó ta truyền thể hiện của lớp này vào trong hàm thiết lập sự kiện tương ứng của View.

Ví dụ nhỏ dưới đây biểu diễn 03 cách thức để tạo và đăng ký sự kiện cho một View mà chúng tôi khuyên dùng.

**Phương pháp 1**: Dùng lớp vô danh (Anonymous Class)

//Tạo một thể hiện của onClickListener()

**private** OnClickListener **mCorkyListener** = **new** **OnClickListener**() {

**public** **void** onClick(View v) {

// Thực hiện các đoạn mã xử lý khi người dùng nhấp vào Button

}

};

**protected** **void** onCreate(Bundle savedValues) {

...

// Lấy Button từ Layout để xử lý

Button button = (Button)findViewById(R.id.corky);

// Đăng ký lắng nghe sự kiện onClickListener cho Button

button.setOnClickListener(**mCorkyListener**);

...

}

**Phương pháp 2**: cho lớp Activity implement Interface Event Listener

**public** **class** ExampleActivity **extends** Activity **implements OnClickListener** {

**protected** **void** onCreate(Bundle savedValues) {

...

Button button = (Button)findViewById(R.id.corky);

button.**setOnClickListener(this);**

}

// Thực hiện phương thức callback của Interface OnClickListener

**public** **void** onClick(View v) {

// Thực hiện các đoạn mã xử lý khi người dùng nhấp vào Button

}

...

}

**Phương pháp 3**: đăng ký sử dụng XML

Phương pháp này được tiến hành thông qua thuộc tính android:onclick trong tập tin xây dựng gia diện XML. Ta chỉ cần đăng ký trong tập tin XML như sau

<Button

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"match\_parent"*

android:text=*"Click me"*

android:onClick=*"onClickButton1"*

android:textAlignment=*"center"*/>

Sau đó trong tập tin Activity ta chỉ cần tạo phương thức có tên là giá trị của thuộc tính android:onClick với tham số là đối tượng View. Trong trường hợp này là onClickButton.

**public** **void** onClickButton1(View v) {

// Thực hiện các đoạn mã xử lý khi người dùng nhấp vào Button

}

Chú ý rằng phương thức callback onClick() ở ví dụ trên thì không có giá trị trả về nhưng một số loại Event Listener khác sẽ trả về một giá trị Boolean.

**Tóm tắt** : Trong phần 2.3 này, ta cần chú ý một số điểm sau :

* Cách thức tạo ra các giao diện người dùng trong Android.
* Sự khác biết giữa các loại ViewGroup mà ta có thể dùng để sắp đặt vị trị cho các View trong giao diện Android.
* Cách tạo giao diện sử dụng phương pháp lập trình hướng đối tượng.
* Bắt và xử lý một số loại sự kiện trên View.

## Thiết kế giao diện người dùng sử dụng đối tượng View

Nội dung chính :

* + - * Sử dụng dụng chế độ Graphic Layout và các View căn bản trong Android để thiết kế giao diện người dùng.
      * Cách bắt sự kiện trên các View căn bản.
      * Cách thức sử dụng Picker View: TimePicker và DatePicker
      * Cách thức sử dụng ListView và GridView để hiện thị danh sách.
      * Cách thức sử dụng nhưng Fragment chuyên dụng.

Ở phần trước chúng tôi đã trình bày về các loại ViewGroup khác nhau mà ta có thể dùng để đặt vị trí cho các View trong Activity. Trong phần này, chúng tôi sẽ trình bày về các loại View khác nhau được dùng cho việc thiết kế giao diện cho ứng dụng Android. Trong Android, các View được chia thành 04 nhóm chính sau đây.

* **Basic View**: Các View được sử dụng thường xuyên như TextView, EditText và Button View.
* **Picker View**: Các View cho phép người dùng đưa ra những chọn lựa từ một danh sách như TimePicker và DatePicker.
* **List View:** Các View trong nhóm này được dùng để hiển thị một danh sách nhiều thành phần như ListView, SpinnerView hoặc GridView.
* **Các Fragment chuyên dụng:** Các Fragment đặc biệt được dùng để thực hiện những chức năng riêng.

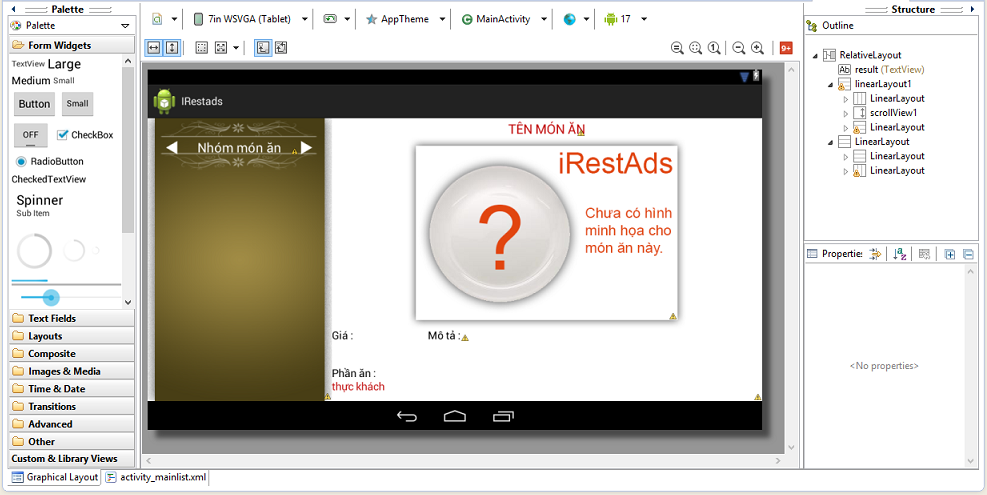
Ở đoạn cuối của phần này sẽ sơ lược về một số loại View khác chưa được để cập đến như một số View về đồng hồ và hiển thị đồ họa. Để bắt đầu phần này, trước tiên chúng tôi sẽ trình bầy về công cụ xây dựng giao diện bằng phương pháp kéo thả được hỗ trợ trong Eclipse được gọi là chế đọ Graphic Layout.

### Chế độ thiết kế giao diện Graphic Layout

Sau khi cài bộ công cụ Android Development Tools (ADT) vào trong Eclipse, ADT sẽ cung cấp nhiều chức năng cho phép ta thiết kế và xây dựng giao diện cho ứng dụng. Trong đó bộ biên tập Graphic Layout chứa nhiều chức năng cho phép ta có thể mở tập tin thiết kế giao diện XML trong Eclipse dưới dạng đồ họa.

Graphic Layout là màn hính chính mà ta sử dụng để thiết kế và xây dựng một cách trực quan, nó được chia thành một số phần sau đây:

* **Canvas**: Vùng nằm chính giữa của màn hình. Hiển thị giao diện được xây dựng từ nội dung của tập tin XML, cho phép ta kéo thả những thành phần giao diện trực tiếp từ phần **Palette.** Ta có thể chọn những phiên bản khác nhau để hiển thị phần Canvas này, với mỗi phiên bản sẽ được mô phỏng theo cách hiển thị thực tế của phiên bản đó. Phiên bản được chọn không cần phải giống với phiên bản được chọn làm mục tiêu phát triển của ứng dụng.
* **Outline:** Vùng nằm ở phía bên phải của màn hình. Vùng này hiển thị cấu trúc của các View trong ứng dụng, ta cũng có thể sắp xếp lại các View. Vùng Outline có những chức năng tương tự Cavas những chỉ hiển thị các View dưới dạng danh sách. Bên dưới vùng này là cửa sổ **Properties** cho phép ta thay đồi những thuộc tính của View.
* **Palette:** Vùng nằm bên trái của màn hình. Nó cung cấp một tập hợp những thành phần mà ta có thể kéo vào trong vùng Canvas. Pelette cũng hiển thị cho ta thấy trước hình dạng của các thành phần hiển thị để dễ dàng cho tìm kiếm thành phần mong muốn.
* **Configuration Chooser:** Nằm ở phía trên cùng của màn hình, nó cung cấp những tùy chọn để thay đổi chế độ màn hình hoặc kiểm màn hình.



Hình ‑ Graphic Layout Editor

**Vùng Canvas và Outline**

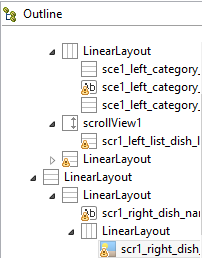
Canvas là khu vực mà ta có thể kéo và thả những thành phần giao diện từ khu Palette để thiết kế giao diện người dùng. Canvas cho phép ta nhìn thấy trước kết quả của giao diện người dùng dựa trên nhiều yếu tố như phiên bản được chọn, chiều màn hình và mẫu giao diện hiện tại thông qua các tùy chọn ở vùng Configuration Chooser. Ngoài ra ta cũng có thể kéo và thả các thành phần vào vùng hiển thị cấu trúc các thành phần trong khu Outline. Vùng Outline này cũng có các chức năng tương tự như Canvas nhưng mục đích chính của nó được dùng để sắp xếp và lựa chọn các thành phần giao diện một cách nhanh chóng. Khi ta nhấp chuột phải vào một thành phần nào đó trên Canvas hoặc Outline, ta sẽ thấy một danh sách các chức năng giúp cho ta thay đổi các thuộc tính của ViewGroup hoặc View được đề cập dưới đây.

* **Một số thuộc tính của View và ViewGroup**

Khi chuột phải vào một View hoặc ViewGroup ta sẽ có thể thay đổi một số thuộc tính như :

* ID của View hoặc ViewGroup.
* Chuỗi hiển thị của View.
* Chiều ngang và chiều cao.
* Các thuộc tính như độ trong suốt (Alpha) hoặc cho phép bắt sự kiện khi người dùng chạm lên View(clickable).
  + - * **Tạo và xem trước các hiệu ứng**

Nếu như View được thiết lập hiệu ứng, ta có thể xem trước hiệu ứng đó trực tiếp trên Canvas bằng chức năng **Play Animation**. Nếu View chưa được gắn hiệu ứng thì ta có thể chọn Create để tạo một hiệu ứng mới.





Hình ‑ Outline

Hình ‑ Canvas

**Vùng Palette**

Chưa những thành phần giao diện ta có thể kéo và thả vào trong vùng Canvas và thêm vào trong giao diện người dùng. Hiển thị các thành phần giao diện theo danh mục và hiển thị trước hình dạng các thành phần để dễ dàng tìm kiếm. Những chức năng chính của Palette gồm có:

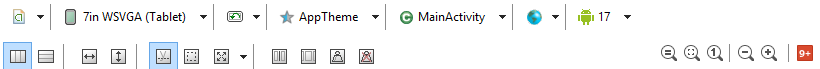
* Những chế độ khác nhau của vùng xem trước: chỉ có Icon, icon và ký tự, tăng giảm kích thước.
* Tùy chỉnh View trong dự án hoặc thư viện bên ngoài và thêm vào trong danh mục Custom Views.
* Sắp xếp thành phần theo thứ tự alphabet.

**Vùng Configuration chooser**

Cho phép ta tạo và cấu hình cho giao diện của ứng dụng trong nhiều tình huống khác nhau. Ví dụ như giao diện khi thiết bị nằm ngàng và giao diện cho thiết bị thẳng đứng.

Ta có thể thiết lập những tùy chọn sau đây cho giao diện:

* Loại màn hình: Đưa ra một số cấu hình về kích thước màn hình dựa trên một số dòng sản phẩm thực tế.
* Chiều màn hình: đặt màn hình nằm ngang hay nằm dọc.
* Chọn chủ đề: chọn một số chủ đề có sẵn hoặc tự tạo chủ để mới.
* Chọn Phiên bản: chọn phiên bản được dùng để hiển thị Canvas và Palette cũng như hiển thị các chủ đề tương ứng.



Hình ‑ Configuration Chooser

### Sử dụng Basic View

Trong phần này chúng tôi sẽ trình bày về một số View căn bản được dùng để thiết kế giao diện :

* TextView và EditText.
* Button và ImageButton.
* CheckBox và ToggleButton
* RadioButton và RadioGroup.

Những View căn bản này cho phép ta hiển thị những thông tin dưới dạng ký tự cũng như một số chức năng lựa chọn căn bản.

**TextView và EditText**

Khi tạo mới một dự án Android, Eclipse luôn tạo sẵn một tập tin main.xml trong thư mục res/layout, tập tin này có sẵn một thành phần <TextView> với nội dung như sau.

<TextView

android:layout\_width=*"wrap\_content"*

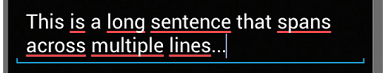
android:layout\_height=*"wrap\_content"*

android:text=*"@string/hello\_world"* />

Như trong ứng dụng HelloWorld đã trình bày ở trước, khi chạy ứng dụng ta sẽ thấy có dòng chữ “Hello World” trên giao diện người dùng, đó là kết quả được hiển thị do thành phần TextView này.

TextView được dùng để hiển thị một chuỗi ký tự cho người dùng. Đây là View căn bản nhất và ta sẽ sử dụng nó thường xuyên trong quá trình phát triển ứng dụng Android. Nếu ta cần cho phép người dùng chỉnh sửa các ký tự được hiển thị thì nên dùng một lớp con của TextView là EditText.

EditText sẽ hiển thị một khu vực hình chữ nhật, nơi mà người dùng có thể nhập những ký tự vào đó. Với thiết lập thuộc tính layout\_height là wrap\_content thì trong trường hợp người dùng nhập một chuỗi ký tự dài thì chiều cao của EditText sẽ được tự động thay đổi để phù hợp và bao trọn được nội dung (Hình 2.4.2-1)

<EditText

Hình ‑ EditText

android:id=*"@+id/txtName"*

android:layout\_width=*"fill\_parent"*

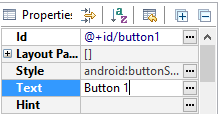
**android:layout\_height=*”wrap\_content”***

/>

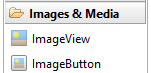
Ta có thể tạo TextView và EditText nhanh chóng bằng cách sử dụng chế độ Graphic Layout. TextView được đặt trong danh mục “Form wigets” và EditText trong danh mục “Text Field” ở đây ta có thể tìm được nhiều loại hiển thị khác nhau như EditText để nhập Password hoặc Email hoặc địa chỉ v...v..

**Button và ImageButton**

Để tạo một Button ta sử dụng Graphic Layout 🡺 danh mục Form wigets 🡺 Kéo thả biểu tượng Button vào trong Canvas 🡺 ở khung Properties tìm đến thuộc tính Text để thay đổi tên của Button được hiển thị.

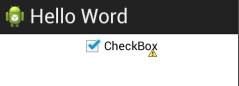
ImageButton có chức năng tương tự như Button nhưng có thêm hình ảnh ở bên trong. Hình ảnh của Button được thiết lập thông qua thuộc tính **src.** Để tạo một ImageButton ta dùng Graphic Layout 🡺 danh mục Image & Media 🡺 Kéo thả biểu tượng ImageButton vào khung Canvas. Khi đó cửa sổ **Resouces Chooser** xuất hiện, chọn hình muốn hiển thị và nhấp OK.

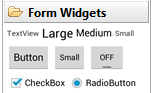
****

**CheckBox và ToggleButton**

CheckBox là một loại Button đặc biệt có hai trạng thái: được đánh dấu và không được đánh dấu. Người dùng chạm vào CheckBox để thay đổi trạng thái của nó.

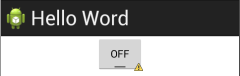
Để tạo một CheckBox ta sử dụng Graphic Layout 🡺 danh mục Form wigets 🡺 Kéo thả biểu tượng CheckBox vào trong Canvas.

****



ToggleButton cũng được dùng như CheckBox, được hiển thị dưới dạng một hình chữ nhật có 02 trạng thái On/Off. Người dùng có thể thay đổi trạng thái của ToggleButton bằng cách nhấp vào nó.

Để tạo một ToggleButton ta sử dụng Graphic Layout 🡺 danh mục Form wigets 🡺 Kéo thả biểu tượng ToggleButton vào trong Canvas.





Cách bắt sự kiện cho CheckBox và ToggleButton là tương tự nhau, đều sự dụng sự kiện OnClickListener và kiểm tra trạng thái isChecked()

CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);

checkBox.setOnClickListener(**new** View.OnClickListener() {

**public** **void** onClick(View v) {

**if** (((CheckBox) v).isChecked()) {

// Thực hiện xử lý khi được đánh dấu

} **else** {

// Thực hiện xử lý khi không được đánh dấu

}

}

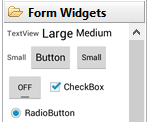
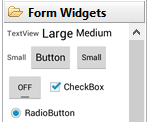
});

**RadioButton và RadioGroup**

RadioButton cũng như CheckBox có hai trạng thái là được đánh dấu và không được đánh dấu. Một RadioGroup được dùng để nhóm một hoặc nhiều RadioButton lại với nhau, và trong đó chỉ có duy nhất một RadioButton có trạng thái được đánh dấu.

Ví dụ dưới đây ta có 02 RadioButton được đặt trong cùng 01 RadioGroup. Thông thường, mỗi RadioButton sẽ hiển thị cho một sự lựa chọn nào đó có nhiều cách chọn, khi người dùng chọn một RadioButton, tất cả những RadioButton còn lại sẽ tự động chuyển về trạng thái không được đánh dấu.

<RadioGroup



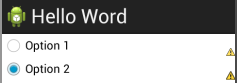
android:id=*"@+id/rdbGp1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

<RadioButton

 android:id=*"@+id/rdb1"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Option 1"* />

<RadioButton

android:id=*"@+id/rdb2"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:checked=*"true"*

android:text=*"Option 2"* />

</RadioGroup>

Cách bắt sự kiện cho RadioButton

RadioGroup radioGroup = (RadioGroup) findViewById(R.id.*rdbGp1*);

radioGroup.setOnCheckedChangeListener(**new** OnCheckedChangeListener() {

**public** **void** onCheckedChanged(RadioGroup group, **int** checkedId) {

RadioButton rb1 = (RadioButton) findViewById(R.id.*rdb1*);

**if** (rb1.isChecked()) {

// Thực hiện xử lý khi RadioButton1 được đánh dấu

} **else** {

// Thực hiện xử lý khi RadioButton1 không được đánh dấu

}

}

});

### Sử dụng PickerView : TimePicker và DatePicker

Trong các ứng dụng viết cho thiệt bị di động như điện thoại và máy tính bảng ta thường gặp phải vấn đề về chọn lựa và thiết lập thời gian để thực hiện một tác vụ nào đó. Android hỗ trợ cho việc giải quyết các vấn đề này thông qua 02 đối tượng View là TimePicker và DatePicker.

**TimePicker**

Là một View được dùng cho việc lựa chọn thời gian trong một ngày, hỗ trợ cả hai cách tính thời gian là theo 24 giờ hoặc chế độ AM/PM.

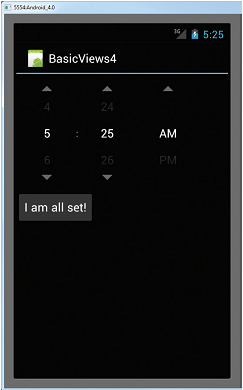
Để trình bày cách sử dụng TimePicker chúng tôi sẽ trình bày thông qua ví dụ sau.

**Bước 1**: Tạo dự án mới có tên là BasicView4

**Bước 2**: Chỉnh sửa lại tập tin main.xml với nội dung như đoạn mã dưới đây

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

 android:layout\_width=*"fill\_parent"*

Hình ‑ TimePicker

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"* >

<TimePicker

android:id=*"@+id/timePicker"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* />

<Button

android:id=*"@+id/btnSet"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:onClick=*"onClick"*

android:text=*"I am all set!"* />

</LinearLayout>

**Bước 3:** Chạy dự án ta sẽ được kết quả hiển thị như hình 2.4.3-1. Sau đó ta có thể nhấp vào các mũi tên ở trên và dưới để tăng hoặc giảm giá trị của TimePicker. Bên cạnh đó ta cũng có thể dùng bàn phím để thiết lập. giá trị cho giờ, phút và giá trị AM/PM.

**Bước 4:** Quay lại Eclipse, mở tập tin BaseView4Activity.java và thay đổi phương thức onCreate() như sau.

TimePicker timePicker;

@Override

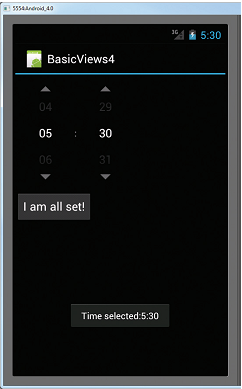
**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

timePicker = (TimePicker) findViewById(R.id.*timePicker*);

timePicker.setIs24HourView(**true**);

}

Hình ‑ TimePicker 24 giờ

**public** **void** onClick(View view) {

Toast.*makeText*(getBaseContext(),

"Time selected:" + timePicker.getCurrentHour()+ ":" + timePicker.getCurrentMinute(),

Toast.*LENGTH\_SHORT*).show();

}

**Bước 5:** Chạy lại ứng dụng ta sẽ thấy TimePicker được hiển thị với định dạng 24 giờ thay vì chế độ AM/PM. Nhấp vào Button sẽ hiển thị thời gian ta đã thiết lập cho TimePicker.

**Giải thích cách thức hoạt động**

TimePicker hiển thị một giao diện cho phép người dùng thiết lập thời gian. Theo mặc định sẽ hiển thị thời gian theo định dạng AM/PM. Nếu muốn chuyển sang định dạng 24 giờ ta phải sử dụng phương thức **setIs24Hour()**. Trong lập trình, để lấy được thời gian mà người dùng thiết lập ta dùng phương thức **getCurrentHour()** và **getCurrentMinute().**

Toast.*makeText*(getBaseContext(),

"Time selected:" + timePicker.getCurrentHour()+ ":" + timePicker.getCurrentMinute(),

Toast.*LENGTH\_SHORT*).show();

*Chú ý :*phương thức getCurrentHour() luôn luôn trả về giờ tính theo định dạng 24 giở.

**DatePicker**

Ngoài TimePicker, Android còn hỗ trợ thêm một View hỗ trợ việc thiết lập thời gian đó là DatePicker. Khi sử dụng DatePicker ta có thể cho người dùng chọn một thời gian cụ thể trong Activity. Thông qua DateActivity ta có thể chọn năm, tháng hoặc ngày tương tự với cách chọn giờ phút trong TimePicker.

Ta cùng xét ví dụ về sử dụng DatePicker sau đây.

**Bước 1:** Sử dụng lại dự án BaseView4 đã tạo ở phần TimePicker, thêm đoạn mã sau vào tập tin main.xml.

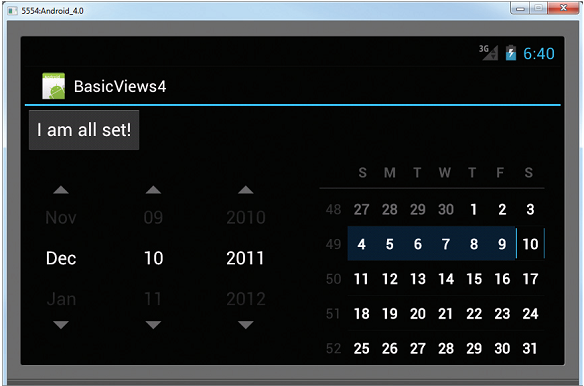
<DatePicker

android:id=*"@+id/datePicker"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"* />

**Bước 2:** Chạy ứng dụng, ta sẽ được kết quả như hình dưới (Chuyển chiều của thiết bị thành nằm ngang).



Hình ‑ DatePicker

**Bước 3:** Quay trở lại Eclipse và chỉnh sửa một số nội dung sau.

Trong tập tin BaseView4Activity.java, thêm đoạn mã sau trong hàm onCreate();

DatePicker datePicker;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

timePicker = (TimePicker) findViewById(R.id.*timePicker*);

timePicker.setIs24HourView(**true**);

// showDialog(TIME\_DIALOG\_ID);

**datePicker = (DatePicker) findViewById(R.id.*datePicker*);**

}

**public** **void** onClick(View view) {

Toast.*makeText*(getBaseContext(),

**"Date selected:" + (datePicker.getMonth() + 1) +**

**"/" + datePicker.getDayOfMonth() +**

**"/" + datePicker.getYear() + "\n" +**

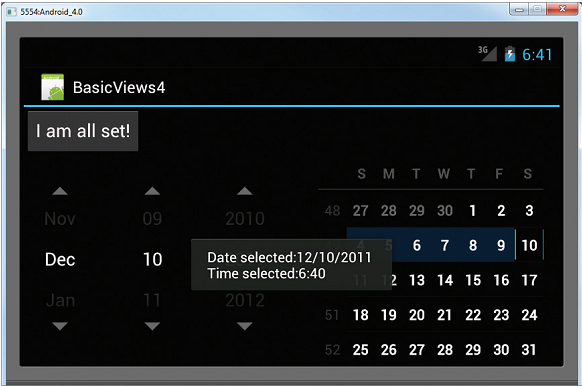
"Time selected:" + timePicker.getCurrentHour() +

":" + timePicker.getCurrentMinute(),

Toast.*LENGTH\_SHORT*).show();

}

**Bước 4:** Chạy lại ứng dụng, sau khi thiết lập thời gian mong muốn nhấp vào Button ta được kết quả hiển thị ngày giờ, thời gian như hình dưới.



Hình ‑ TimePicker và DatePicker

**Giải thích cách thức hoạt động**

Giống như TimePicker, ta gọi phương thức getMonth(), getDayOfMount() và getYear() để lấy giá trị về tháng, ngày, nằm được người dùng thiết lập.

**"Date selected:" + (datePicker.getMonth() + 1) +**

**"/" + datePicker.getDayOfMonth() +**

**"/" + datePicker.getYear()**

*Chú ý:* phương thức getMonth có giả trị trả từ 0 đến 11 với 0 là tháng một, 1 là tháng 2 v..v.. Do đó ta cần tăng kết quả của phương này lên lên 1 đơn vị để lấy tháng tương ứng khi hiển thị dưới dạng số.

### Sử dụng ListView và GridView

Các View danh sách là những View cho phép ta hiển một danh sách dài những phần tử. Trong Android, có hỗ trợ 02 loại View danh sách: **ListView** và **SpinnerView.** Cả hai View này đều rất hữu dụng cho việc hiển thị danh sách các thành phần, nhưng trong báo cáo này chúng tôi chỉ đề cập đến **ListView**.

Ngoài hiển thị dưới dạng danh sách, trong các ứng dụng có hiển thị nhiều hình ảnh hoặc có tính chất trưng bày sản phẩm thì một kiểu hiển thị hay được sử dụng nữa đó là hiển thị dưới dạng lưới. Trong Android ta thường xử dụng đối tượng **GridView** để thực hiện việc hiển thị này.

**ListView**

ListView là một loại ViewGroup hiển thị danh sách các phần tử View thành phần. Những phần tử của ListView được thêm tự động thông qua việc sử dụng một Adapter, Adapter này lấy nội dung từ một nguồn như một mảng hoặc từ một câu truy vấn cơ sử dữ liệu và chuyển đổi các phần tử trong đó vào một View và đặt nó vào trong danh sách. Có nhiều cách để hiệu chỉnh giao diện cho ListView và GridView nhưng chúng tôi chỉ trình bày cách thực hiện dễ dàng và canh bản nhất với mục địch giới thiệu về 02 loại View này.



Hình ‑ Bố cục của ListView

Cùng tìm hiểu ví dụ sau để hiểu về cách sử dụng ListView.

**Bước 1:** Tạo dự án Android có tên BaseView5.

**Bước 2:** Mở tập tin string.xml trong thư mục res/values và thêm nội dung sau đây.

<string-array name=*"presidents\_array"*>

<item>Dwight D. Eisenhower</item>

<item>John F. Kennedy</item>

<item>Lyndon B. Johnson</item>

<item>Richard Nixon</item>

<item>Gerald Ford</item>

<item>Jimmy Carter</item>

<item>Ronald Reagan</item>

<item>George H. W. Bush</item>

<item>Bill Clinton</item>

<item>George W. Bush</item>

<item>Barack Obama</item>

</string-array>

**Bước 3:** Mở tập tin BaseView5Activity.java và chỉnh sửa lại phương thức onCreate().

**public** **class** BasicViews5Activity **extends ListActivity** {

String[] presidents;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

//setContentView(R.layout.main);

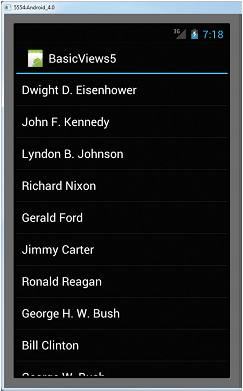
presidents = getResources().getStringArray(R.array.*presidents\_array*);

**setListAdapter(new ArrayAdapter<String>(this, android.R.layout.*simple\_list\_item\_1*, presidents));**

}

**public** **void** onListItemClick(ListView l, View v, **int** position, **long** id) {

Toast.*makeText*(**this**, "You have selected " + presidents[position], Toast.*LENGTH\_SHORT*).show();

}

}

**Bước 4:** Chạy ứng dụng và ta thấy trên giao diện sẽ hiển thị danh sách tên của các tổng thống nước Mỹ (Hình 2.4.4-2).

**Bước 5:** Nhập vào một phần tử của danh sách sẽ xuất hiện một thông báo chứa nội dung của phần tử được chọn.

**Giải thích cách thực hiện**

Đầu tiên, việc thêm vào trong tập tin strings.xml một thẻ **<string-array>** là để khai báo thêm một tài nguyên nữa cho ứng dụng với kiểu là mảng chuỗi ký tự. Với cách khai báo như trên ta có thể lấy giá trị của chuỗi trên qua phương pháp lập trình bằng cách sử dụng phương thức getResources() trong tập tin BaseView5Activity.java.

getResources().getStringArray(R.array.*presidents\_array*);

Hình ‑ ListView đơn giản

Nhìn chung, ta có thể lấy tất cả các tài nguyên đi kèm trong ứng dụng bằng cách sử dụng phương thức getResources().

Điều cần chú ý tiếp them là trong ví dụ này, lớp BasicView5Activity được kế thừa từ lớp ListActivity chứ không phải Activity như thường lệ. ListActivity là một lớp được mở rộng từ lớp Activity, nó hiển thị một danh sách các phần tử bằng cách kết nối với một nguồn dữ liệu nào đó. Cũng cần chú ý thêm, trong ví dụ này ta không cần chỉnh sử tập tin main.xml để chứa ListView, tự bản thân lớp ListActivity đã chứa sẵn một ListView trong đó. Do đó, trong phương thức onCreate(), ta không cần gọi đến hàm setContentView() để tải giao diện từ tập tin main.xml.

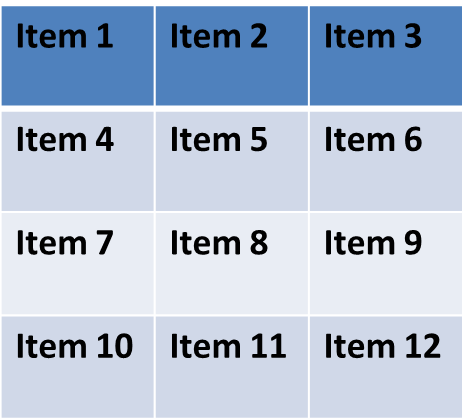
Thay vào đó, trong phương thức onCreate(), ta sử dụng phương thức setListAdapter() để lập trình cho việc đổ đầy giao diện của Activity với một ListView. Sau đó dùng đối tượng ArrayAdapter để quản lý mảng các chuỗi ký tự sẽ được hiển thị. Trong ví dụ này, ta thiết lập cho ListView hiển thị theo chế độ đơn giản nhất : simple\_list\_item\_1.

**setListAdapter(new ArrayAdapter<String>(this, android.R.layout.*simple\_list\_item\_1*, presidents));**

* Ngoài việc hiển thị một danh sách các phần tử đơn giản ta cũng có thể thiết lập cho ListView thêm những loại hiển thị khác như có hộp kiểm hoặc tự thiết kế riêng cách hiển thị cho phần tử.

**GridView**

GridView cũng là một loại ViewGroup cho phép hiển thị các phần tử ở cả hai chiều. Những phần tử của lưới sẽ được thêm tự động vào giao diện bằng cách dùng đối tượng ListAdapter.



Hình ‑ Bố cục của GridView

Ví dụ sử dụng GridView

**Bước 1:** Tạo dự án Android mới đặt tên là GridView.

**Bước 2:** Mở tập tin main.xml và sửa nội dung lại như sau

<GridView xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/gridView1"*

android:numColumns=*"auto\_fit"*

android:gravity=*"center"*

android:columnWidth=*"50dp"*

android:stretchMode=*"columnWidth"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"* >

</GridView>

**Bước 3:** Sửa tập tin GridViewActivity.java với đoạn mã sau

GridView gridView;

**static** **final** String[] *numbers* = **new** String[] {

"A", "B", "C", "D", "E",

"F", "G", "H", "I", "J",

"K", "L", "M", "N", "O",

"P", "Q", "R", "S", "T",

"U", "V", "W", "X", "Y", "Z"};

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.main);

gridView = (GridView) findViewById(R.id.*gridView1*);

**ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,**

**android.R.layout.*simple\_list\_item\_1*, *numbers*);**

**gridView.setAdapter(adapter);**

gridView.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> parent, View v,

**int** position, **long** id) {

Toast.makeText(getApplicationContext(),

((TextView) v).getText(), Toast.LENGTH\_SHORT).show();

}

});

}

**Bước 4:** Chạy ứng dụng ta được màn hình hiển thị như hình dưới. Sau đó nhấp vào các chứ cái sẽ hiển thị thông báo với nội dung của chữ cái đó.



Hình ‑ Ứng dụng GridView biểu diẽn 24 chữ cái

**Giải thích cách thức hoạt động**

Tương tự như ListView, GridView cũng có khả năng tự động phát sinh các phần tử dựa trên một nguồn dữ liệu (Trong trường hợp này là một mảng chuỗi ký tự). Do đó ta tạo một đối tượng ArrayAdapter với kiểu hiển thị là dạng đơn giản simple\_list\_item\_1 và truyền vô dữ liệu muốn hiển thị là danh sách 24 chữ cái.

**ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,**

**android.R.layout.*simple\_list\_item\_1*, *numbers*);**

Sau đó ta tiếp tục gắn ArrayAdapter cho GridView bằng cách sử dụng phương thức sau:

**gridView.setAdapter(adapter);**

Để bắt sự kiện khi người dùng chọn phần từ ta sử dụng Interface OnItemClickListener.

gridView.setOnItemClickListener(**new** OnItemClickListener() {

**public** **void** onItemClick(AdapterView<?> parent, View v,**int position**, **long** id) {

Toast.makeText(getApplicationContext(),((TextView) v).getText(), Toast.LENGTH\_SHORT).show();

}

});

Trong phương thức onItemClick có truyền vào tham số int position. Tham số này được dùng để chỉ vị trí của phần tử được chọn.

## Xử lý, hiển thị hình ảnh và sử dụng Menu

Nội dung chính:

* Cách thức sử dụng những View thông dụng để hiển thị hình ảnh: ImageView, ImageSwicher, Gallery và hiển thị theo dạng GridView.
* Tạo và sử dụng Options Menu và Context Menu.

Ở phần trước, ta đã được học về các loại View khác nhau có thể được dùng để xây dựng giao diện cho ứng dụng Android. Trong phân này, ta sẽ tiếp tục tìm hiểu về các loại View khác giúp ta xây dựng một ứng dụng mạnh mẽ và hấp dẫn hơn. Đó là các View có chức năng hiển thị hình ảnh và Menu.

Trong tài liệu báo cáo này, từ lúc bắt đầu đến này tất cả những View đều được sử dụng để hiện thị thông tin dưới dạng ký tự. Để hiển thị một hình ảnh, ta có thể dùng **ImageView**, **Gallery**, **ImageSwitcher** hoặc **GridView**.

Vì ứng dụng được thực hiện trong đề tài không sử dụng Gallery và GridView đã được trình bày ở phần trước nên chúng tôi chỉ trình bày chi tiết về 02 loại View còn lại.

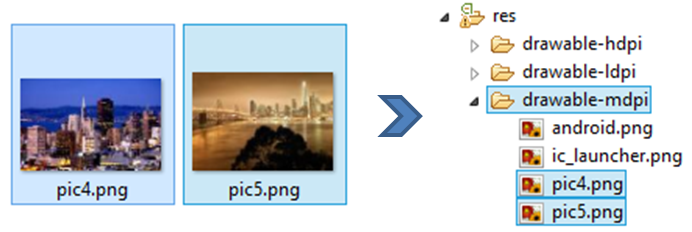
### ImageView

Cho phép ta hiển thị những hình ảnh tùy ý, lớp ImageView có thể tải hình ảnh lên từ nhiều nguồn khác nhau, ta cũng có thể tùy chỉnh kích thước của ImageView bằng cách dùng các LayoutParamater và dùng các chế độ hiển thị như phóng to và thu nhỏ hình.

Cùng thực hiện ví dụ sau để tìm hiểu cách sử dụng ImageView.

**Bước 1:** Tạo một dự án Android mới đặt tên là ImageView.

**Bước 2:** Chọn một số hình ảnh muốn hiển thị và copy chúng vào thư mục res/ drawable-mdpi. Ở đây chúng tôi chọn 02 hình pic4.png và pic5.png.



Hình ‑ Thêm hình ảnh cho dự án Android

**Bước 3:** Mở tập tin res/layout/main.xml và thêm vào nội dung sau.

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"300dp"*

android:layout\_height=*"200dp"*

android:scaleType=*"fitXY"*

android:src=*"@drawable/pic4"* />

<Button

android:id=*"@+id/btnChangeImage"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:text=*"Change Image"* />

**Bước 4:** Sửa nội dung tập tin ImageViewActivity.java như sau

Button button;

ImageView image;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

image = (ImageView) findViewById(R.id.*imageView1*);

button = (Button) findViewById(R.id.*btnChangeImage*);

button.setOnClickListener(**new** OnClickListener() {

@Override

**public** **void** onClick(View arg0) {

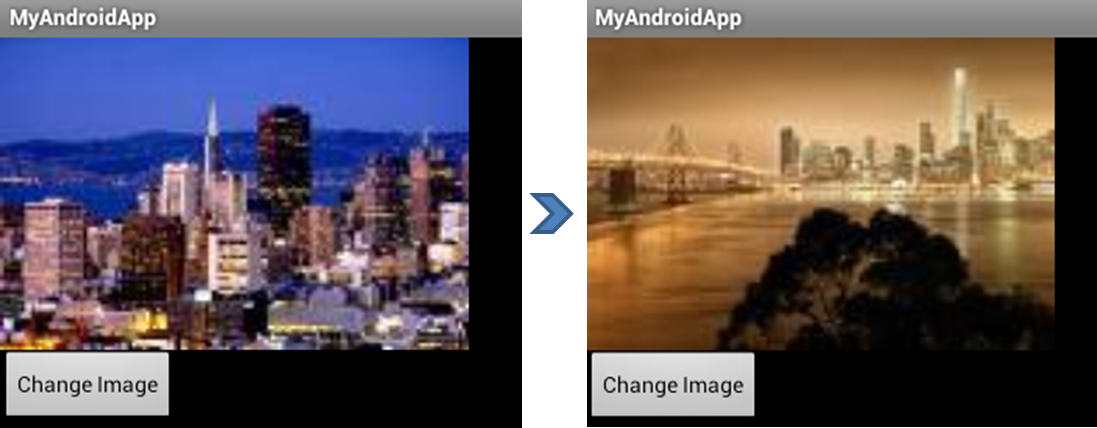
image.setImageResource(R.drawable.*pic5*);

}

});

}

**Bước 5:** Chạy ứng dụng, sau đó nhấp vào nút “Change Image” ta sẽ thấy hình ảnh hiển thị sẽ được thay đổi.



Hình ‑ Thay đổi hình ảnh ImageView bằng phương pháp lập trình

**Giải thích cách thức hoạt động**

Trong Android, thư mục **res** được dùng để lưu trữ và quản lý các tài nguyên được sử dụng trong ứng dụng. Các tài nguyên đó có các tập tin XML hoặc tập tin hình ảnh. Để một hình ảnh bên ngoài trở thành tài nguyên của ứng ta phải đưa nó vào trong các thư mục drawable-xyz với là nhưng thư mục tương ứng với loại màn hình của thiết bị.

Trong Android đơn vị được dùng để xác định loại màn hình được gọi là “dot per inch (dpi)” để chỉ số lượng điểm ảnh trên một inch.

Trong Android có hỗ trợ sẵn 04 loại kích thước màn hình sau:

* Low density (ldpi) : Mật độ thấp khoảng 120 dpi.
* Medium density (mdpi): Mật độ trung bình khoảng 160 dpi.
* Hight density (hdpi): Mật độ cao khoảng 240 dpi.
* Extra hight density (xhdpi): Mật độ dày đặc khoảng 320 dpi.

Tùy thuộc vào loại thiết bị mà ta sẽ đặt hình ảnh vào với thư mục drawable tương ứng. Sau khi đã cho hình ảnh vào thư mục drawable ta có thể truy xuất đến hình ảnh thông qua đối tượng quản lý tài nguyên R và phương thức setImageResource().

image.setImageResource(R.drawable.*pic5*);

Để sử dụng ImageView ta khai báo trong tập tin XML như sau

<ImageView

android:id=*"@+id/imageView1"*

android:layout\_width=*"300dp"*

android:layout\_height=*"200dp"*

android:scaleType=*"fitXY"*

android:src=*"@drawable/pic4"* />

Đối tượng ImageView được quy định hình ảnh thông qua thuộc tính android:src và có thể được thay đổi kích thước bằng thuộc tính layout\_width và layout\_height như những View khác. Điểm đặc biệt là cơ chế tự động co giãn hình hay còn được gọi là scalling. Trong ví dụ này chúng tôi sử dụng 02 ảnh với kích thước 135x89 pixel. Nhưng khi chạy ứng dụng ta thấy ảnh đã được đổ đầy kích thước được quy định là 300x200. Để làm được điều đó ta phải quy định chế độ giãn hình bằng thuộc tính android:scaleType= “fitXY”.

* Một số cách thức thiết lập hình ảnh cho ImageView bằng phương pháp lập trình:

**setImageResoure (int resId) :** thiết lập hình ảnh cho ImageView sử dụng tài nguyên của ứng dụng được đặt trong thư mục res/drawable-xyz.

**setImageBitmap (Bitmap bm) :** thiết lập một đối tượng hình ảnh Bitmap làm nội dung cho ImageView. Chi tiết về Bitmap sẽ được trình bày ở đoạn cuối của phần này.

**setImageDrawable (Drawable drawable) :** thiết lập một đối tượng Drawable làm nội dung cho ImageView.

**setImageAlpha(int alpha) :** thiết lập giá trị **Alpha** sẽ được áp dụng cho hình ảnh. Giá trị Alpha trong Anroid tương tự thuộc tính **opacity** của CSS trong lập trình Web. Thuộc tính này quy định độ trong suốt của hình ảnh có giá trị trong khoảng 0 🡺 1.

### ImageSwicher

ImageSwicher có thể được hiểu như một danh sách các hình ảnh liên tiếp nhau với những hiệu ứng khi thay đổi hình ảnh. ImageSwicher thường được sử dụng để chuyển đổi những hình ảnh bằng cách chạy các hiệu ứng trong quá trình chuyển đổi. Phần này sẽ hướng dẫn cách sử dụng mã Java để tạo các hiệu ứng cho ImageSwicher cũng như tạo hiệu ứng bằng tập tin XML trong thư mục res/anim.

Cùng tìm hiểu cách sử dụng ImageSwicher qua ví dụ sau.

**Bước 1:** Tạo một ứng dụng Android mới đặt tên là ImageSwicher

**Bước 2:** Chọn một số hình ảnh cần hiển thị và đặt vào thư mục res/drawable-xyz.



**Bước 3:** Mở tập tin main.xml và thêm nội dung sau.

<ImageSwitcher

android:id=*"@+id/imageswitcher"*

android:layout\_width=*"300dp"*

android:layout\_height=*"200dp"*

android:layout\_margin=*"10dp"* />

**Bước 4:** Mở tập tin ImageSwicherActivity.java và lần lượt sửa đổi như sau.

Khai báo lớp ImageSwicherActivity thực hiện Interface **ViewFactory**. Sau đó ta sẽ phải thực hiện lại phương thức **makeView()** cần phải thực hiện của Interface ViewFactory.

**public** **class** MainActivity **extends** Activity **implements ViewFactory** {

@Override

**public** View **makeView()** {

// **TODO** Auto-generated method stub

**return** **null**;

}

}

Sau đó ta thêm vào các đoạn mã sau

**private** Integer[] imageIds = { R.drawable.*pic2*, R.drawable.*pic3*,

R.drawable.*pic4*, R.drawable.*pic5*,

R.drawable.*pic6*, };

ImageSwitcher imageSwitcher;

**int** imageIndex = 0;

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

imageSwitcher = (ImageSwitcher) findViewById(R.id.*imageswitcher*);

imageSwitcher.setFactory(**this**);

Animation fadeIn = AnimationUtils.*loadAnimation*(**this**, android.R.anim.*slide\_in\_left*);

fadeIn.setDuration(3000);

Animation fadeOut = AnimationUtils.*loadAnimation*(**this**, android.R.anim.*slide\_out\_right*);

fadeOut.setDuration(3000);

imageSwitcher.setInAnimation(fadeIn);

imageSwitcher.setOutAnimation(fadeOut);

imageSwitcher.setImageResource(imageIds[imageIndex]);

}

Tiếp theo sửa lại phương thức makeView() và thêm một phương thức sử lý sự kiện onClick() như sau.

@Override

**public** View makeView() {

ImageView i = **new** ImageView(**this**);

i.setScaleType(ImageView.ScaleType.*FIT\_CENTER*);

i.setClickable(**true**);

i.setOnClickListener(clickItem);

i.setLayoutParams(**new** ImageSwitcher.LayoutParams(LayoutParams.*WRAP\_CONTENT*, LayoutParams.*WRAP\_CONTENT*));

**return** i;

}

**public** OnClickListener clickItem = **new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

imageSwitcher.setImageResource(imageIds[imageIndex]);

imageIndex++;

**if** (imageIndex > (imageIds.length - 1)) {

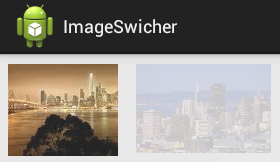
imageIndex = 0;

}

}

};

**Bước 5:** Chạy ứng dụng, chờ 03 giây ta sẽ thầy hình ảnh đầu tiên hiển thị, sau đó nhấp vào ảnh đó sẽ thấy hiệu ứng thay đổi hình ảnh chạy qua bên phải và xuất hiện hình ảnh mới.

**Giải thích cách thức hoạt động**

Hình ‑ ImageSwicher

Điều đầu tiên cần chú ý trong ví dụ này là lớp ImageSwicherActivity không chỉ được mở rộng từ lớp Activity mà còn hiện thưc Interface ViewFactory. Để áp dụng Interfac này cho ImageSwicher ta dùng phương thức sau.

imageSwitcher.setFactory(**this**);

Interface này cho phép ta tạo một đối tượng View mà ImageSwicher sẽ sử dụng. Nội dung của View được tạo được thực hiện trong phương thức makeView(). Trong ví dụ này ta tạo ra một đối tượng ImageView cho phép người dùng chạm lên và thực hiện việc thay đổi hình ảnh.

ImageView i = **new** ImageView(**this**);

i.setClickable(**true**);

i.setOnClickListener(clickItem);

Đối tượng ImageSwicher sẽ thay đổi hình ảnh hiển thị thông qua những phương thức thiết lập tương tự ImageView. Trong trường hợp này chúng tôi sử dụng phương thức lấy lên từ tài nguyên của ứng dụng : imageSwitcher.setImageResource(imageIds[imageIndex]);

Để tạo hiệu ứng xuất hiện và biến mất cho ImageSwicher ta sử dụng đối tượng thực hiện các hiệu ứng Animation.

Animation fadeIn = AnimationUtils.*loadAnimation*(**this**, android.R.anim.*slide\_in\_left*);

Đối tượng Animation được tạo lên từ đối tượng AnimationUtils thông qua phương thức loadAnimation() sẽ tải hiệu ứng được định nghĩa trong thư mục res/anim của hệ điều hành Android. Ta có thể lựa chọn loại hiệu ứng có sẵn hoặc tự tạo cho mình hiệu ứng riêng bằng cách tạo một thự mục tên anim trong thư mục res. Sau đó tạo tiếp một tập tin XML để định nghĩa hiệu ứng của riêng mình (trong phạm vi tài liệu này chúng tôi không trình bày chi tiết về cách tạo tập tin hiệu ứng XML).

fadeIn.setDuration(3000);

Ta cũng có thể thiết lập độ dài thời gian mà hiệu ứng thực hiện tính bằng milisecond

imageSwitcher.setInAnimation(fadeIn);

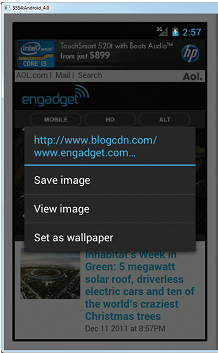
imageSwitcher.setOutAnimation(fadeOut);

Gắn hiệu ứng vào cho ImageSwicher, có 02 loại hiệu ứng đó là hiệu ứng xuất hiện và hiệu ứng biến mất (In animation và Out animation ).

### Sử dụng Menu trong ứng dụng Android

Hình ‑

Menu là một thành phần giao diện thường được sử dụng trong nhiều ứng dụng thuộc các lĩnh vực khác nhau. Menu được dùng để thêm các thao tác và lựa chọn cho người dùng trong khi tương tác với các Activity. Điều đặc biệt là nó không hiển thị trực tiếp trên giao diện chính mà chỉ được kích hoạt khi có tác động của người dùng. Có 02 loại Menu chinh trong Android.

* Options Menu: Hiển thị thông tin có liên quan đến Activity hiện tại. Trong Android, ta kích hoạt Options Menu bằng cách ấn vào phím MENU trên Action Bar.
* Context Menu: Hiển thị thông tin có liên quá đến một thành phần View trong Activity. Trong Android ta kích hoạt Context Menu bằng cách chạm vã giữ trên thành phần giao diện đó.

Hình ‑

Hình 2.5.3-1 là một ví dụ cho Options Menu trong ứng dụng trình duyệt Web. Options Menu được hiển thị khi người dùng nhấn vào nút MENU. Những phần tử của Menu được thay đổi phù hợp với Activity hiện tại.

Hình 2.5.3-2 biểu thị một Context Menu, được hiện thị khi người dùng chạm và giữ một hình ảnh được hiển thị trên trang Web.

**Tạo các phương thức trợ giúp**

Trước khi bắt đầu tìm hiểm và tạo 02 loại Menu đã được trình bày ở trên, ta cần tạo 02 phương thức giúp đỡ cho chúng.

Một phương thức được dùng để tạo các phần tử được hiển thị bên trong Menu, và một để xử lý các sự kiện xảy ra khi người dùng chọn một phần tử của Menu.

**Bước 1:** Tạo một dự án Android mới đặt tên là Menus.

**Bước 2:** Mở tập tin MenusActivity.java và thêm vào 02 phương thức sau

**private** **void** createMenu(Menu menu) {

MenuItem mnu1 = **menu.add(0, 0, 0, "Item 1");**

MenuItem mnu2 = **menu.add(0, 1, 1, "Item 2");**

MenuItem mnu3 = **menu.add(0, 2, 2, "Item 3");**

}

**private** **boolean** menuChoice(MenuItem item) {

**switch** (item.getItemId()) {

**case** 0:

Toast.*makeText*(**this**, "Click Item 1", Toast.*LENGTH\_LONG*).show();

**return** **true**;

**case** 1:

Toast.*makeText*(**this**, "Click Item 2", Toast.*LENGTH\_LONG*).show();

**return** **true**;

**case** 2:

Toast.*makeText*(**this**, "Click Item 3", Toast.*LENGTH\_LONG*).show();

**return** **true**;

}

**return** **false**;

}

**Giải thích cách thức hoạt động**

Phương thức createMenu() nhận vào tham số là một đối tượng Menu và thêm một dãy các phần tử cho Menu này. Để thêm một phần tử cho Menu, ta tao một thể hiện của lớp MenuItem và dùng phương thức add() của đối tượng Menu.

**menu.add(0, 0, 0, "Item 1");**

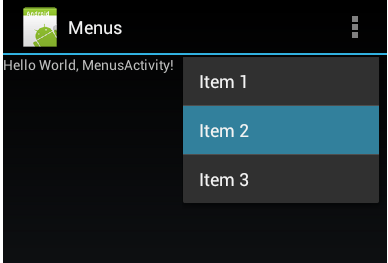
Phương thức add(int groupId, int itemId, int order, String title) nhận vào 04 tham số có ý nghĩa lần lượt như sau.

* groupId: Số định danh nhóm cho các phần tử, sử dụng giá trị 0 nếu phần tử không ở trong nhóm nào.
* itemId: chỉ số định danh cho phân tử là duy nhất trong Menu.
* order: thứ tự mà phần tử được hiển thị.
* title: Ký tự hiển thị của phần tử.

Phương thức menuChoice() nhận vào một tham số là đối tượng Menu và kiểm tra giá trị định danh của phần tử được chọn. Sau đó sẽ hiển thị một thông điệp để người dùng biết phần tử nào đã được chọn.

### Options Menu

Chúng tôi sẽ trình bày về cách thức để thực hiện một Options Menu trong Android thông qua ví dụ sau đây.

**Bước 1:** Sử dụng lại dự án Menus đã tạo ở phần trước. Mở tập tin MenusActivity.java và thêm vào 02 phương thức sau đây.

Hình ‑Options Menu

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

**super**.onCreateOptionsMenu(menu);

createMenu(menu);

**return** **true**;

}

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**return** menuChoice(item);

}

**private** **void** createMenu(Menu menu) {

...

}

**private** **boolean** menuChoice(MenuItem item) {

...

}

**Bước 2:** Chạy ứng dụng và nhấp vào nut MENU trên thanh Action Bar ta sẽ thấy hiển thị danh sách các phần tử của Options Menu. Nhấp vào một phần tử bất kỳ sẽ xuất hiện thông báo.

**Giải thích cơ chế hoạt động**

Để hiển thị Options Menu cho Activity, ta cần thực hiện 02 phương thức trong Activity là onCreateOptionMenu() và onOptionsItemSelected().

* onCreateOptionMenu() được gọi khi người dùng ấn nút MENU, trong ví dụ này sẽ gọi đến phương thức trợ giúp createMenu() để thêm các phần tử cho Menu.
* onOptionsItemSelected() được gọi đến khi người dùng nhấp lên phần tử của Menu. Trong ví dụ ta gọi đến phương thức menuChoice() để hiển thị nội dung của phần tử được chọn.

### Context Menu

Trong phần Options Menu đã trình bày làm thế nào để hiển thị một Options Menu khi người dùng nhấp vào nút MENU. Bên cạnh Options Menu, ta cũng có thể hiển thị một loại Menu nữa là Context Menu. Một Context Menu thường được sử dụng kết hợp với một thành phần View trong Activity. Context Menu sẽ được hiển thị khi người dùng chạm và giữ một một tử. Trong ví dụ dưới đây, nếu người dùng chạm thành phần Button và giữ khoảng vài giây, một Context Menu sẽ được hiển thị. Để một View có thể kích hoạt được Context Menu ta cần gọi phương thức setOnCreateContextMenuListener() cho thành phần giao diện đó. Đoàn mã dưới đây sẽ trình bày cách thức kết hợp Context Menu với một Button.

**Bước 1:** Sử dụng lại dự án đã Menus đã được tạo ở mục trước. Thêm một Button vào nội dung của tập tin main.xml

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"Click and hold on it"* />

**Bước 2:** Thay đổi tập tin MenusActivity.java như sau.

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

Button btn = (Button) findViewById(R.id.*button1*);

btn.setOnCreateContextMenuListener(**this**);

}

@Override

**public** **void** onCreateContextMenu(ContextMenu m, View v, ContextMenuInfo mI) {

**super**.onCreateContextMenu(m, v, mI);

createMenu(menu);

}

@Override

**public** **boolean** onContextItemSelected(MenuItem item) {

**return** menuChoice(item);

}

**private** **void** createMenu(Menu menu) {

...

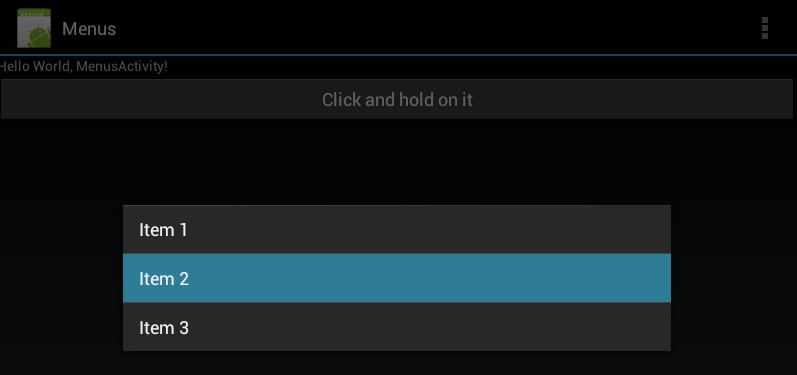
}

**private** **boolean** menuChoice(MenuItem item) {

...

}

**Bước 3:** Chạy ứng dụng, sau đó chạm và giữ Button ta sẽ thấy xuất hiện Context Menu như hình 2.5.5-1



Hình ‑ Context Menu

**Giải thích cách thức hoạt động**

Như đã nói ở phần đầu, ta gọi phương thức setOnCreateContextMenuListener() của thành phần Button để liên kết nó với một Context Menu. Khi người dùng nhấp và giữ Button, phương thức onCreateContextMenu() sẽ được gọi. Trong phương thức này, ta gọi đến phương thức createMenu() để tạo các phần tử cho Context Menu. Tương tự như Options Menu, khi một phần tử của Menu được chọn phương thưucs onContextItemSelected() được gọi và phương thức menuChoice() tiến hành xử lý.

## Xử lý tác vụ AsyncTask, Timer và Handler

Nội dung chính :

* + - * Làm quen với sử lý tác vụ bất đồng bộ AsynTask.
      * Làm quen với bộ định thời Timer và Handler

Ở chương này chúng tôi trình bày với mục đích cung cấp khái niệm và các tính năng của các đối tượng AsynTask, Timer và Handler. Việc sử dụng các đối tượng này sẽ được trình bày lồng vào các phần khác hoặc để giải quyết một vấn đề sẽ được đề cập trong chương 4 của tài liệu này.

### Tác vụ bất đồng bộ AsyncTask

AsyncTask là đối tượng giúp cho việc sử dụng các tiến trình xử lý giao diện dễ dàng và hợp lý hơn. Đối tượng này cho phép tiến hành các hoạt động nền và hiển thị kết quả trên giao diện trong khi hệ thống đang sử lý những tác vụ khác.

Một tác vụ bất đồng bộ được định nghĩa là một chuỗi các phép tính được thực hiện trong một tiến trình ngầm và kết quả của nó được hiển thị trên giao diện thông qua một tiến trình khác. Một tiến trình bất đồng bộ còn được định nghĩa bởi 03 loại kiểu dữ liệu chung được gọi là **Params, Progress, Result** cùng với 04 bước là **onPreExecute, doInBackground, onProcessUpdate và onPostExecute**.

**Cách sử dụng**

Để sử dụng đối tượng AsyncTask ta phải thông qua một lớp con được kế thừa từ lớp trừu tượng AsyncTask, lớp con này sẽ viết lại các phương thức trừu tượng của AsyncTask trong đó bắt buộc phải viết lại phương thức **doInBackground(Params…)** và phương thức **onPostExecute(Result).**

**public** **class** ExampleAsyncTask **extends AsyncTask<Integer, String, Boolean>** {

@Override

**protected** **void** onPreExecute() {

// **TODO** Auto-generated method stub

**super**.onPreExecute();

}

@Override

**protected** Boolean doInBackground(Integer... arg0) {

// **TODO** Auto-generated method stub

**return** **null**;

}

@Override

**protected** **void** onProgressUpdate(String... values) {

// **TODO** Auto-generated method stub

**super**.onProgressUpdate(values);

}

@Override

**protected** **void** onPostExecute(Boolean result) {

// **TODO** Auto-generated method stub

**super**.onPostExecute(result);

} }

Đoạn mã trên là một ví dụ về các phương thức trong một đối tượng AsyncTask. Sau khi đã kế thừa AsyncTask, để sử dụng ta tạo một thể hiện của lớp ExampleAsyncTask() và gọi hàm execute() như sau.

**new** ExampleAsyncTask().execute(1,2,3);

Để hiểu ý nghĩa của các tham số truyền vào như đoạn mã trên, chúng tôi sẽ trình bày về 03 kiểu dữ liệu chung định nghĩa một AsyncTask.

**03 kiểu dữ liệu chung của AsyncTask**

Để tạo một AsyncTask ta cần chỉ định 03 kiểu dữ liệu được dùng cho ba mục đích sau đây trong một Asynctask.

**public** **class** ExampleAsyncTask **extends** AsyncTask<Params, Progress, Result> {

Khi sử dụng AsyncTask, trong trường hợp ta cần xử lý dữ liệu đầu vào và trả về một kết quả nào đó khi hoàn thành tác vụ thì ta sẽ phải truyền các dữ liệu đầu vào thông qua hàm execute() của AsyncTask.

* **Params**, kiểu dữ liệu của các tham số được gửi để thực hiện tác vụ.
* **Progress**, kiểu dữ liệu của những đơn vị tiến trình được trả lại trong quá trình tính toán.
* **Result**, kiểu dữ liệu trả về của quá trình tính toán.

Trong trường hợp không cần xử lý kiểu dữ liệu nào ta sử dụng đối tượng **Void** của Java để định nghĩa AsyncTask.

**04 giai đoạn được thực hiện trong một AsyncTask**

Khi một tác vụ bất đồng bộ được tiến hành, tác vụ đó sẽ trải qua 04 giai đoạn sau đây :

1. **onPreExecute**(): gọi đến trước khi tác vụ được tiến hành. Giai đoạn này thường được dùng để thiết lập tác vụ,ví dụ như hiển thị một thông báo hoặc một thanh tiến trình trên giao diện.
2. **doInBackground**(Params…): được gọi trong một tiến trình ngầm ngay sau khi việc thực thi phương thức onPreExecute() hoàn tất. Giai đoạn này được dùng để thực hiện những phép tính có thể diễn ra trong thời gian dài. Những tham số của tác vụ bất đồng bộ được chuyển đến giai đoạn này để thực hiện các tính toán xử lý. Kết quả của việc xử lý sẽ trở thành giá trị trả về của giai đoạn này và tiếp tục được chuyển đến gia đoạn cuối cùng onPostExecute().

Giai đoạn này cũng có thể sử dụng phương thức **publishProgress** (Progress…) để cập nhật trạng thái của tác vụ lên giao diện thông qua giai đoạn thứ ba của tác vụ là **onProgressUpdate**(Progress…).

1. **onProgressUpdate**(Progress…) được gọi trong một tiến trình xử lý giao diện sau khi phương thức publishProgress() được gọi. Phương thức này được dùng để hiển thị trạng thái của tiến trình lên giao diện người dùng trong khi quá trình tính toán ngầm vẫn được thực thi.
2. **onPostExecute (**Result**)** được gọi trong một tiến trình giao diện khi quá trình tính toàn ngầm hoàn tất. Kết quả của việc tính toàn được chuyển đến giai đoạn này như một tham số đầu vào.

**Dừng tác vụ bất đồng bộ**

Một tác vụ có thể bị dừng bất cứ lúc nào bằng cách gọi đến phương thức **cancel(boolean)**. Khi phương thức này được gọi đến nó sẽ tiến hành một chuỗi các lời gọi hàm khác. Đầu tiên sẽ gọi đến hàm **isCancelled(),** sau đó sẽ gọi đến hàm onCancelled thay vì hàm **onPostExecute.** Trong trường hợp ta muốn việc dừng tác vụ diễn ra nhanh nhất có thể, ta nên kiểm tra giá trị của hàm isCancelled tước khi tiến hành một đơn vị tính toán trong phương thức **doInBackground.**

### Bộ định thời Timer vàm TimerTask

Đối tượng Timer là đối tượng dược Java hỗ trợ cho phép tạo lịch trình để thực hiện một số tác vụ nào đó. Mỗi Timer có một tiến trình để thực hiện một chuỗi các tác vụ, khi tiến trình này đang xử lý một tác vụ thì những tác vụ còn lại sẽ được thiết lập một độ trễ nhất định (delay). Có hai chế độ để Timer thực hiện tác vụ :

* Chế độ One-shot: được lên lịch để thực hiện tác vụ tại một thời điểm xác định hoặc sau một độ trễ về thời gian nào đó.
* Chế độ Recurring: các tác vụ được lên lịch thực sau một khoảng thời gian (**fixed-period**) hoặc theo một tỷ lệ thời gian (**fixed-rate**) nhất đinh.

Khi một Timer không còn cần thiết nữa ta sẽ gọi phương thức **cancel()**, phương thức này sẽ giải phóng những tiến trình và tài nguyên của Timer.

Đối tượng Timer này không đảm bảo về vấn đề thời gian thực của việc lên lịch trình cho các tác vụ. Nhiều tiến trình có thể thực hiện trong cùng một thời gian mà không cần phải tiến hành quá trình đồng bộ.

**Một số phương thức thường dùng của đối tượng Timer**

* Public void **cancel()**

Hủy Timer và tất cả nhưng tác vụ đã được lên lịch. Nếu đang có một tác vụ đang chạy thì tác vụ đó sẽ không bị ảnh hưởng. Sauk hi đã hủy sẽ không thể lên lịch cho thêm tác vụ nào nữa.

* Public int **purge()**

Xóa tất cả những tác vụ đã bị hủy khỏi hàng đợi. Trả về số lượng các tác vụ đã bị hủy và được xóa khỏi hàng đợi

* Public void **schedule(TimerTask** task, **Date** when, **long** period**)**

Lên lịch cho một tác vụ được lập lại sau một thời gian được chỉ định.

Tham số:

* Task: tác vụ cần thực hiện.
* When: thời gian thực hiện lần đầu tiên
* Period: khoảng thời gian dãn cách giữa 02 lần thực hiện
* Public void **schedule(TimerTask** task, **long** delay, **long** period**)**

Lên lịch cho một tác vụ được lặp lại sau một thời gian được chỉ định.

Tham số:

* Task: tác vụ cần thực hiện
* Delay: khoảng thời gian phải chờ trước khi thực hiện tác vụ lần đầu tiên.
* Period: khoảng thời gian dãn cách giưa 02 lần thực hiện.
* Public void **schedule(TimerTask** task, **Date** when**)**

Lên lịch để thực hiện tác vụ một lần

#### public void schedule ([TimerTask](file:///D:\_Working\Program%20Files\Android%204.0\docs\reference\java\util\TimerTask.html) task, [Date](file:///D:\_Working\Program%20Files\Android%204.0\docs\reference\java\util\Date.html) when)

Added in [API level 1](file:///D:\_Working\Program%20Files\Android%204.0\docs\guide\topics\manifest\uses-sdk-element.html#ApiLevels)

Schedule a task for single execution. If **when** is less than the current time, it will be scheduled to be executed as soon as possible.

##### Parameters

|  |  |
| --- | --- |
| *task* | the task to schedule. |
| *when* | time of execution. |

# NỘI DUNG NGHIÊN CỨU: LIFERAY PORTAL

## Những điều cần biết về Liferay Portal

### Giới thiệu về Liferay Portal

Liferay Portal là mã nguồn mở (open source) và miến phí được viết bằng Java và được phân phối theo giấy phép của GNU Lesser General Public License và giấy phép độc quyền. Nó được sử dụng mạnh mẽ trong mạng nội bộ của công ty (intranets) và mạng diện rộng (extranets).

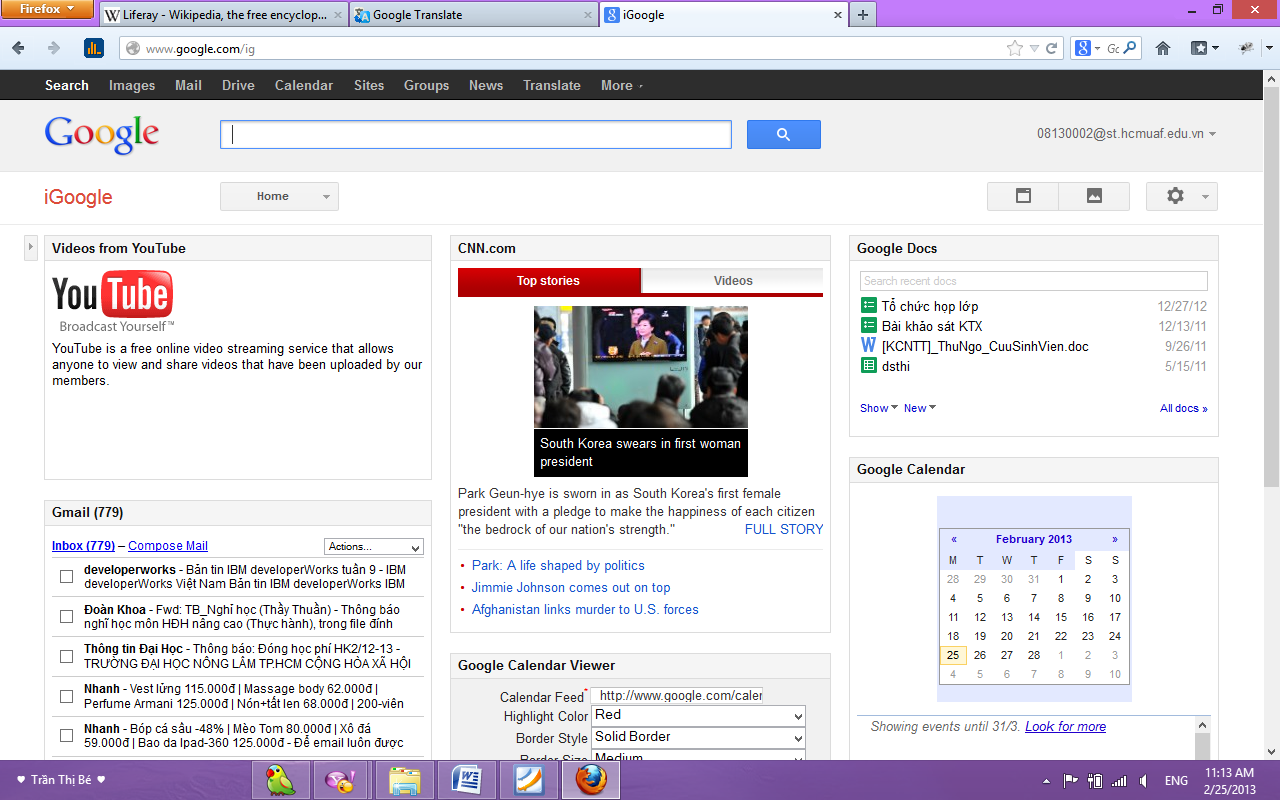
Liferay Portal cho phép người dùng thiết lập các chức năng phổ biến của một trang web. Về cơ bản là xây dựng các đơn vị chức năng gọi là porlet.Liferay Portal đôi lúc cũng được mô tả như một framework quản lý nội dung hay là một framework cho ứng dụng web(web application). Liferay hỗ trợ cho các plugins mở rộng sang nhiều ngôn ngữ lập trình, bao gồm hỗ trợ cho portlet PHP và Ruby.

Liferay Portal là dựa trên Java và chạy trên bất kỳ nền tảng điện toán có khả năng chạy Java Runtime Environment và một máy chủ ứng dụng. Liferay có sẵn bản đi kèm với một container servlet như là Apache Tomcat.

### Portal là gì

Portal là một tập hợp các ứng dụng web mini (mini web applications), gọi là porlets. Một portal hỗ trợ các tính năng như quản lý thông tin cá nhân (persionalization), tập hợp nội dung (content aggregation), chứng thực (atuthentication) và tùy biến (customization). Các porlet hoạt động như là những cửa sổ trên ứng dụng web bên trong portal, với mỗi cửa sổ như vậy trên trang web của portal (portal page) đại điện cho một porlet.

Để có được một cảm nhận rõ hơn về về portal, bạn có thể truy cập vào trang iGoogle (<http://www.google.com/ig>). Hình bên dưới cho thấy một trang chủ iGoogle portal sau khi một người dùng đã đăng nhập. Bạn có thể thấy các portlet hiển thị email từ Gmail, tiêu đề từ CNN, nội dung từ YouTube và nhứng cái khác tương tự như thế. Các porlet có thể thay đổi thông tin hiển thị của email hoặc thay đổi số lượng tiều đề của CNN mà họ muốn xem….Hơn thế nữa người dùng có thể kéo để sắp xếp các cửa sổ của porlet đi bất cứ đâu trên trang portal theo ý muốn. Đồng thời học có thể thêm nhiều hơn các porlet hoặc xóa bỏ chúng ra khỏi trang của portal (portal page).



Hình ‑ Trang chủ của iGoogle gốm nhiều porlet thể hiện thông tin từ các nguồn khác nhau.

*Chú ý: Porlet được gọi là ứng dụng web mini (web application mini) vì chúng cung cấp ít hơn các thông tin và chức năng cho người dùng so với ứng dụng web ban đầu mà mà nó thể hiện. Ví dụ ở hình 3.1.2-1 trên thì porlet Gmail hiển thị email và cung cấp các tùy chọn để soạn hoặc xóa các email, nhưng nó khôngcung cấp một tùy chọn để thêm email vào danh sách công việc của bạn, được cung cấp bởi ứng dụng Gmail trên web gốc.*

### Lợi ích của web portal

Web portals cung cấp các lợi ích từ các nhóm ứng dụng web khác nhau. Người dùng cuối có thể truy cập các nguồn thông tin khác nhau trên các ứng dụng web khác nhau từ một trang web portal. Ví dụ, một người sử dụng có thể truy cập nội dung từ các nguồn khác nhau bằng cách trực tiếp truy cập vào các ứng dụng web khác nhau, hoặc bằng cách truy cập những ứng dụng web từ một trang web mạng nội bộ, hoặc bằng cách sử dụng một sự kết hợp của các web portal và các ứng dụng web (web application).

### Porlet là gì

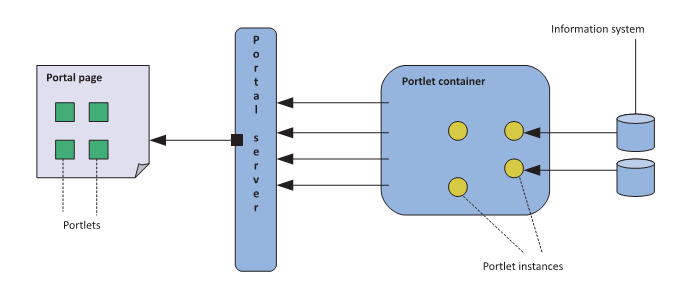
Porlet là một thành phần giao diện người dùng có khả năng gắn kết, nó cung cấp nội dung một cách cụ thể, mà có thể là một dịch vụ hoặc thông tin từ các hệ thống thông tin hiện có. Các porlet cung cấp giao diện người dùng của portal bằng cách truy cập vào các ứng dụng riêng biệt, các hệ thống, hoặc các nguồn dữ liệu để hiển thì thành các phần nội dung khác nhau tới trang web portal của người dùng. Một ví dụ về porlet như là porlet về tin tức cung cấp tin tức mới nhất khi truy cập một Yahoo.

### Tại sao sử dụng porlet

Porlet đại diện cho các dịch vụ và là các thành phần có khả năng tích hợp, bạn có thể thêm vào, thay đổi trạng thái, hành vi hoặc làm đa dạng ứng dụng của bạn bằng cách sử dụng porlet. Bởi vì các porlet có khả năng tương tác lẫn nhau tại tầng giao diện người dùng nên nó có vai trò rất quan trọng trong việc phát triển các ứng dụng. Bên cạnh đó với khả năng tích hợp của mình, porlet còn rất hữu ích đối với việc phát triển các ứng dụng từ các dịch vụ có sẵn.

Một trong những tính năng quan trọng nhất của portlet là liên portlet giao tiếp,được xây dựng vào kiến ​​trúc portlet.

### Cơ sở hạ tầng của portal

****

Hình ‑ Cơ sở hạ tầng của portal

Cơ sở hạ tầng của portal (portal infrastructure) bao gốm một portal server và một porlet container. Porlet container quản lý các thể hiện của porlet và cách hiện thị các porlet trên trang portal.

## Cài đặt Liferay

### Yêu cầu về hệ thống

* Để cài đặt Liferay Portal thì ổ cứng cần phải còn trống nhiều hơn 300MB , bộ nhớ tạm thời (RAM ) tối thiểu là 1GB và một cổng (port) trống cho server

### Cài đặt Liferay Tomcat bundle

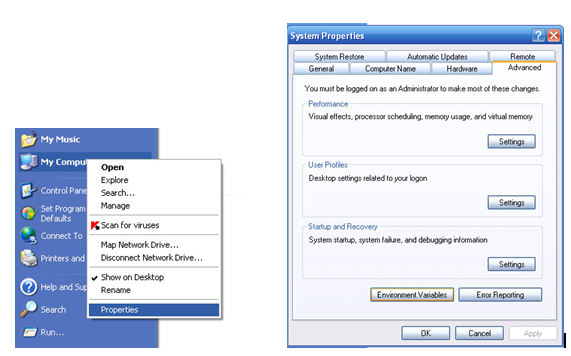
**Cài đặt Java và cấu hình biến môi trường cho Java**

1. [Download JDK](http://www.oracle.com/technetwork/java/javase/downloads/index.html) và cài đặt JDK
2. Thiết lập các biến môi trường sau:

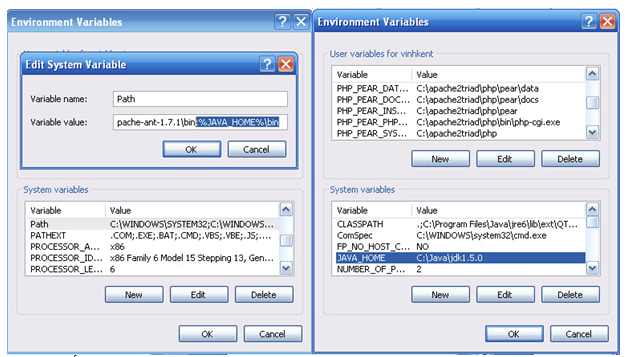
**JAVA\_HOME = {java\_dir}\jdk1.6.0.x**

**Path = {java\_dir}\jdk1.6.0.x\bin**

1. Đối với JAVA\_HOME thì ta tạo mới trong phần **System Variables**, còn biến Path thì thêm vào biến Path có sẵn và phân cách bằng dấu “**;**”



Hình ‑ Chọn biến môi trường.

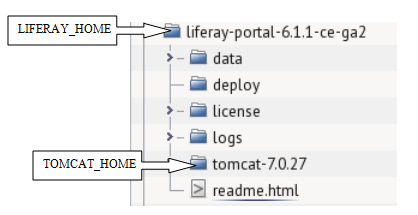


Hình ‑ Thiết lập giá trị cho biến môi trường.

**Tiến hành cài đặt Liferay Tomcat bundle**

Để cài đặt Liferay Tomcat bundle bạn làm theo các bước:

1. Tải phiên bản Liferay Tomcat bundle mới nhất tại trang web: <http://www.liferay.com/downloads/liferay-portal/available-releases>
2. Giải nén gói vừa tải được vào một thư mục nào đó (theo ví dụ hình dưới thì thư mục đó có tên là **bundles**) và ta có được thư mục **LIFERAY\_HOME** và **TOMCAT\_HOME** như hình vẽ dưới :

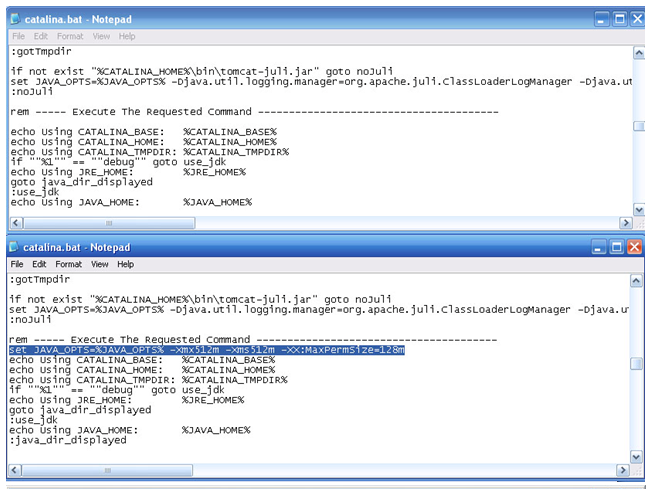


Hình ‑ Cấu trúc thư mục truong Liferay Portal

1. Vào **TOMCAT\_HOME/bin** và nhấp đôi vào **startup.bat**.

Nếu trong quá trình khởi động server mà không xuất hiện cửa sổ server “Tomcat”, bạn cần thêm 1 dòng định nghĩa sau vào file “**catalina.bat**”

**set JAVA\_OPTS=%JAVA\_OPTS% -Xmx512m -Xms512m -X:MaxPermSize=128m**

****

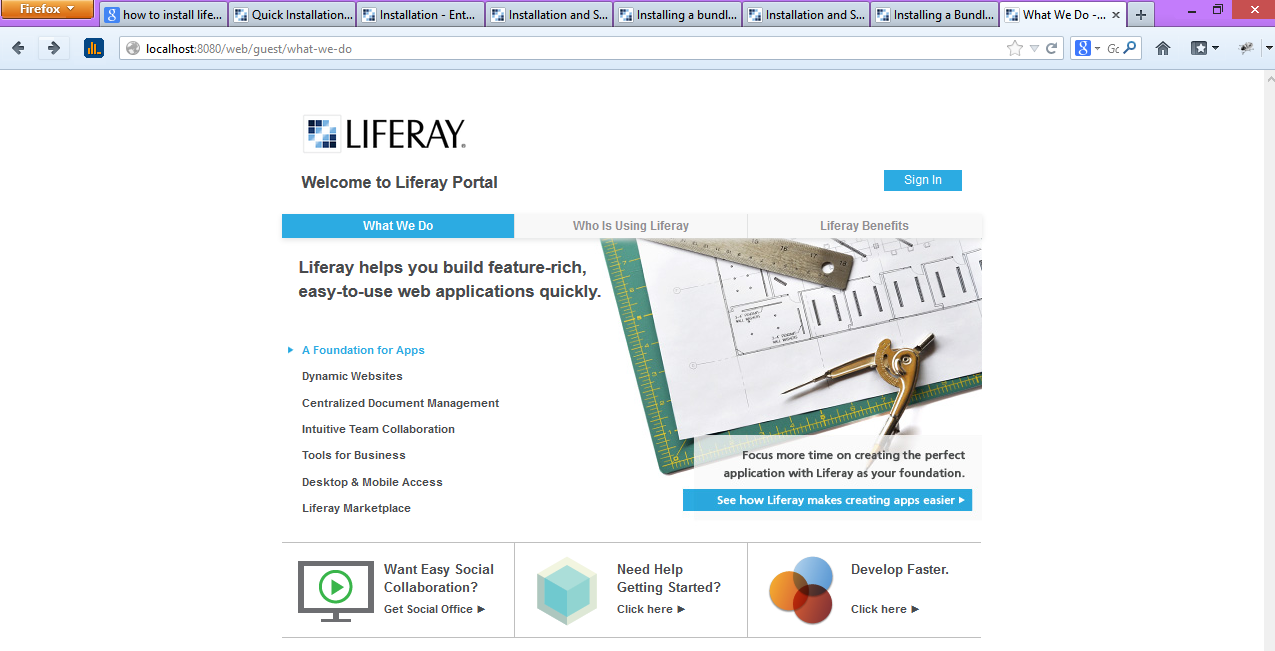
Hình ‑ Cấu hình thâm số cho tập tin catalina.bat

1. Server sẽ khởi động ngay lập tức và xuất hiện cửa sổ sau:



Hình ‑Server khởi động thành công.

1. Server khởi động thành công sẽ hiện thị giao diện trang chủ của liferay như sau :



Hình ‑Trang chủ Liferay

## Tích hợp Liferay Portal vào Eclipse

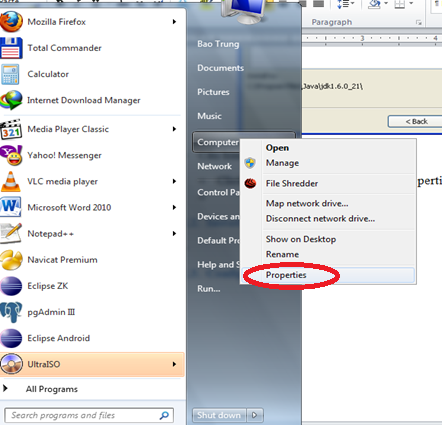
### Cài đặt Ant

Bước 1: [Download Apache Ant](http://ant.apache.org/bindownload.cgi).

Bước 2: Giải nén Ant.

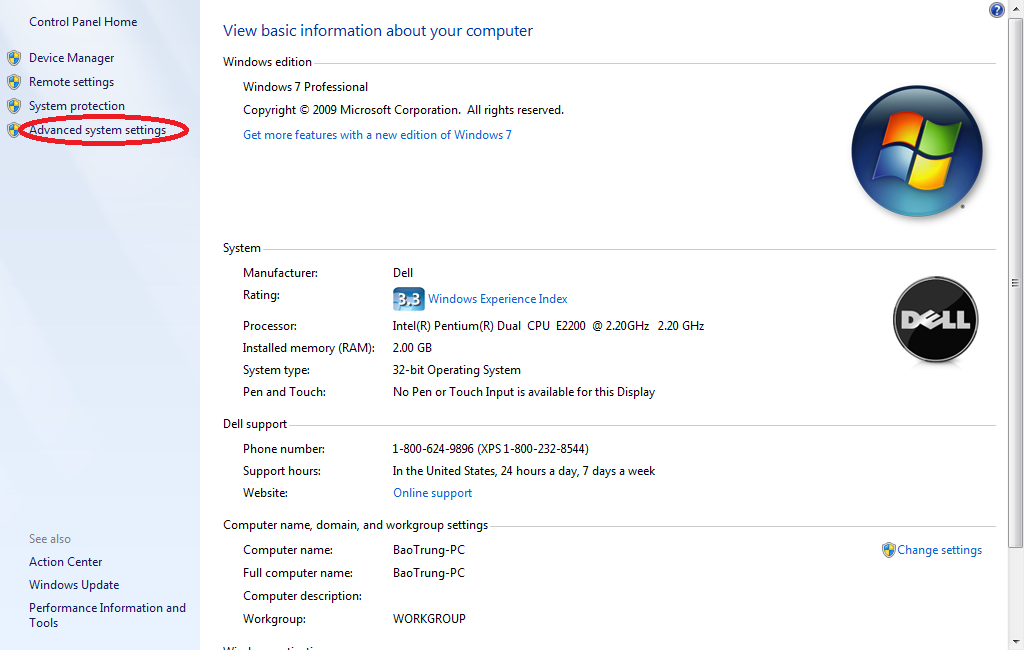
Bước 3: Cấu hình biến môi trường:

Bước 3.1: Click phải vào my computer chọn properties:



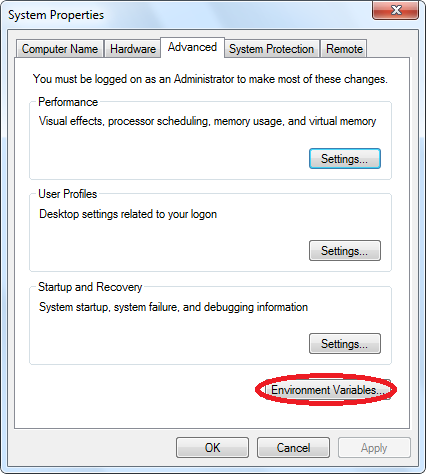
Hình ‑ Cấu hình biến môi trường Ant 1

Bước 3.2: Chọn “Advanced system setting” trong cửa sổ properties.



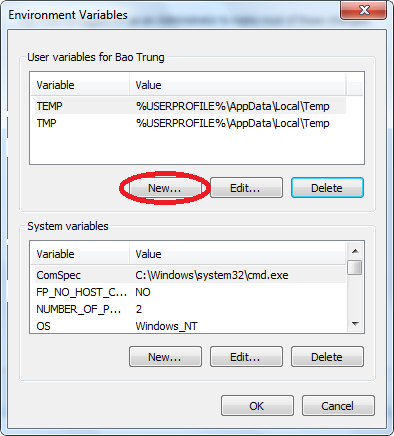
Hình ‑ cấu hình biến môi trường Ant 2

Bước 3.3: Click chọn button “Enviroment Variables”



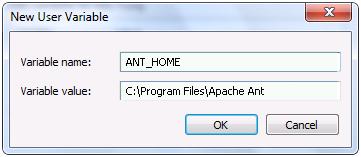
Hình ‑ Cấu hình biến môi trường Ant 3

Bước 3.4: Cửa sổ “Enviroment variables” xuất hiện:



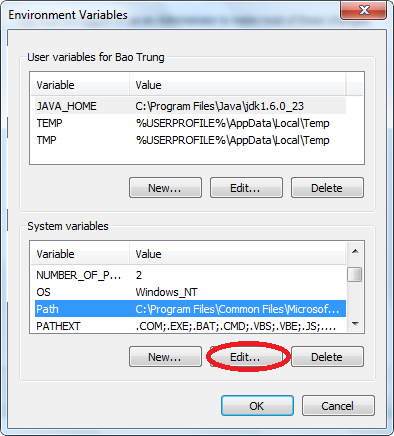
Hình ‑ Cấu hình biến môi trường Ant 4

Bước 3.5: Chọn button “new…” để thêm mới biến môi trường. Nhập tên biến là ANT \_HOME và value là đường dẫn đến thư mục Ant.



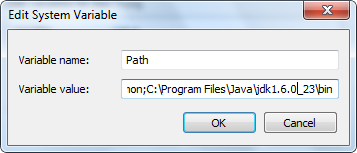
Hình ‑ Vấu hình biến môi trường Ant 5

Bước 3.6: Tìm đến biến path và chọn edit.



Hình ‑ Cấu hình biến môi trường Ant 6

Bước 3.7: Thêm vào value giá trị “;” và đường dẫn đến thư mục bin trong Ant.



Hình ‑ Cấu hình biến môi trường Ant 7

### Cài đặt eclipse

Bước 1: [Download Eclipse](http://www.eclipse.org/downloads/).

Bước 2: Giải nén eclipse ra một thư mục.

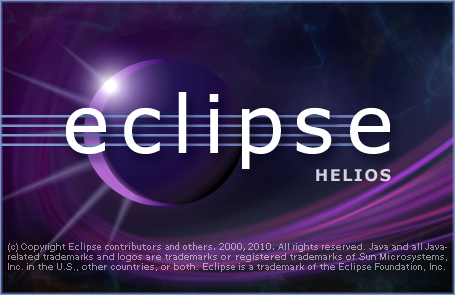
Bước 3: Chạy file eclipse.exe để khởi động eclipse.

### Cài đặt và cấu hình liferay plugin SDK trong eclipse

Bước 1: [Download Liferay Plugin SDK](http://www.liferay.com/downloads/liferay-portal/available-releases).

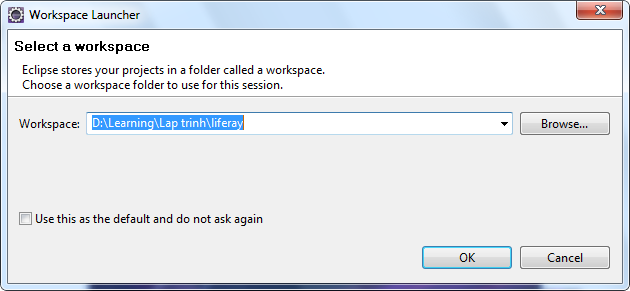
Bước 2: Giải nén Liferay Plugin SDK ra một thư mục.

Bước 3: Chạy eclipse.



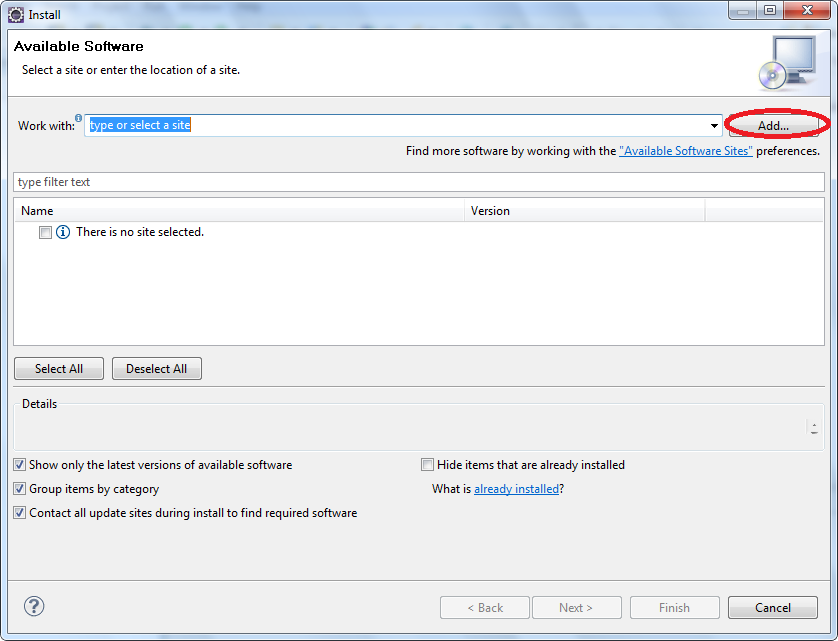
Hình ‑ Eclipse flash screen

Bước 4: Chọn workspace làm việc và click ok.



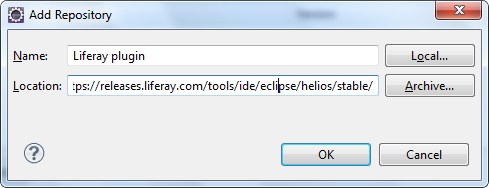
Hình ‑ Select eclipse workspace

Bước 5: Trong giao diện chính của eclipse chọn menu “Help>Install New Software…”. Cửa sổ cài đặt plugin của eclispe sẽ xuất hiện, bạn click “Add” để thêm plugin mới:



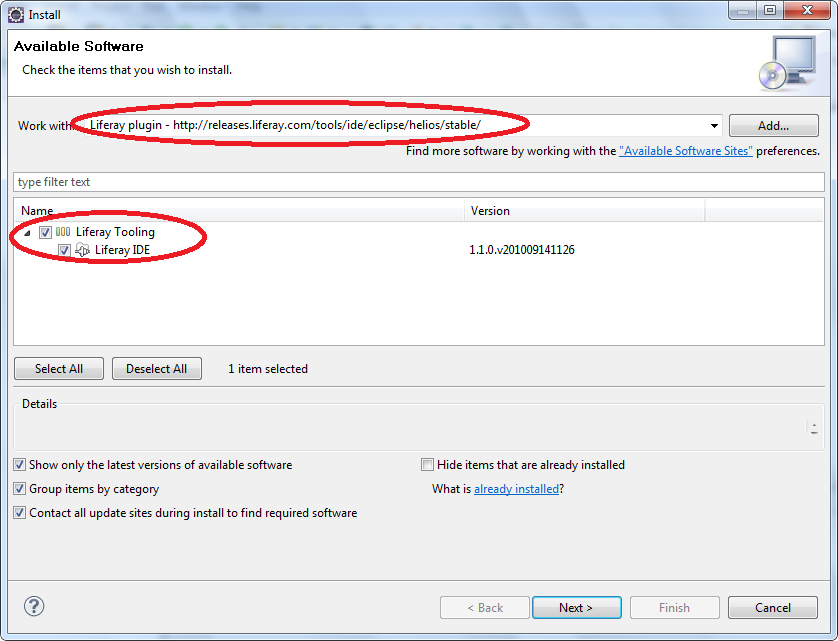
Hình ‑ Install new eclipse plugin

Bước 6: Trong hộp thoại hiện ra bạn nhập name là “Liferay plugin” và location là <http://releases.liferay.com/tools/ide/eclipse/helios/stable/>. Sau đó click “OK”.



Hình ‑ Add new eclipse plugin site

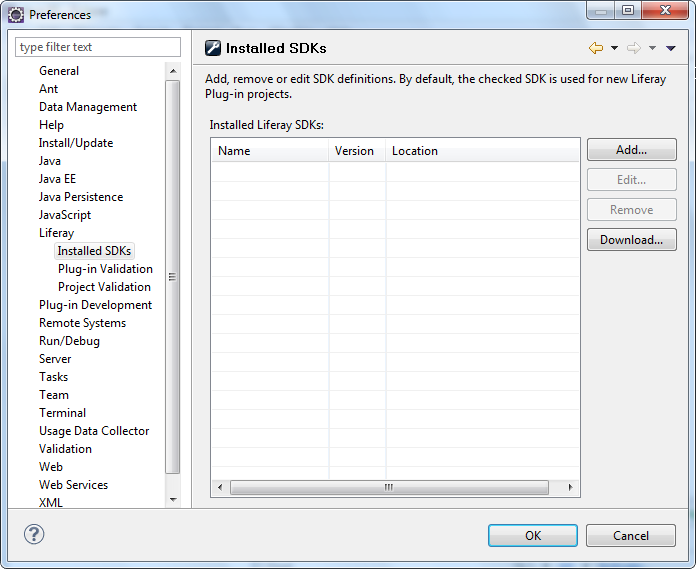
Bước 7: Chọn “Liferay plugin” từ combobox “Work with”, sau đó chọn “Liferay tooling”.



Hình ‑ Install Liferay plugin SDK

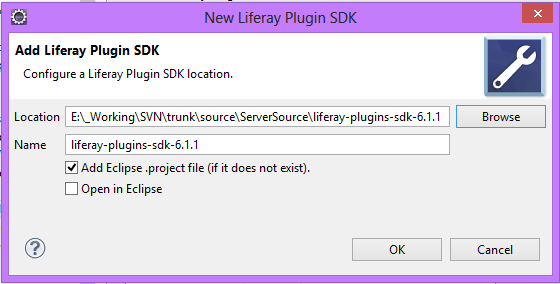
Bước 8: Click “Next” và chọn accept để hoàn tất cài đặt. Sau khi cài đặt hoàn tất khởi động lại eclipse để plugin có hiệu lực.

Bước 9: Trong eclipse chọn menu “Window🡪Prefernces”. Trong cửa sổ hiện ra chọn “Liferay🡪Installed SDKs” và click button “Add” ở bên phải.



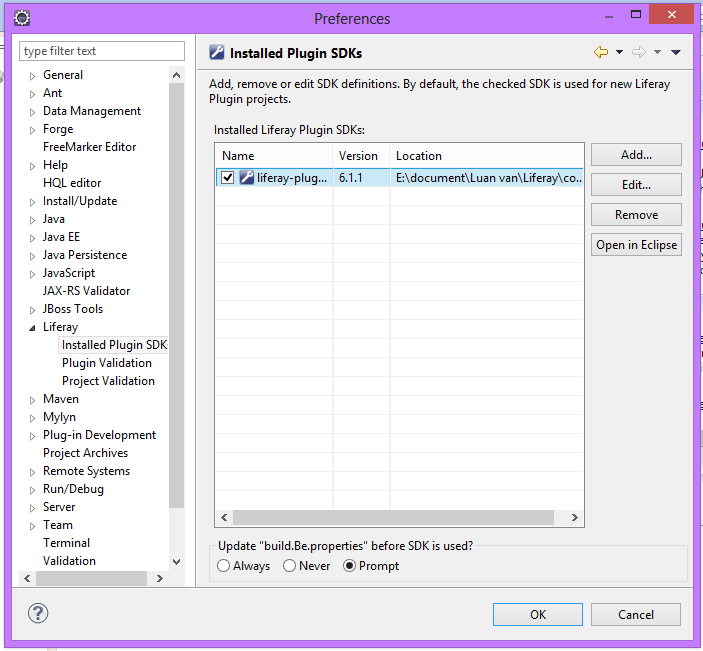
Hình ‑ Liferay SDK configuration 1

Bước 10: Tại trường “Location” chọn đến thư mục giải nén SDK và click “OK”.



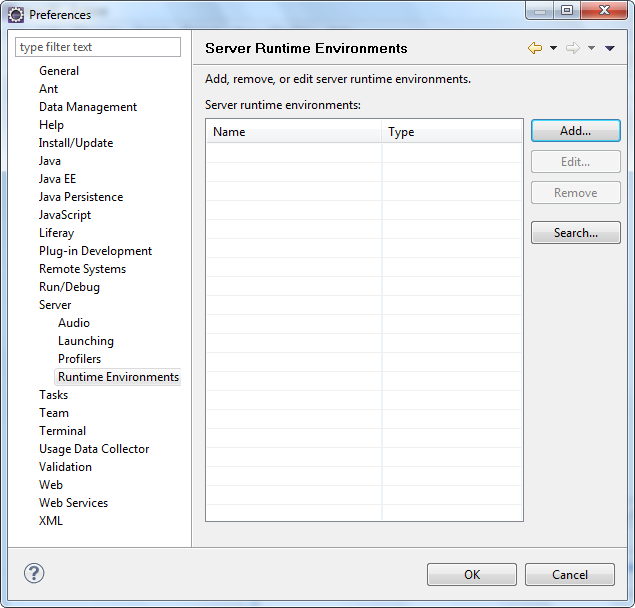
Hình ‑ Liferay SDK configuration 2

Bước 11: Chọn checkbox liferay plugin.



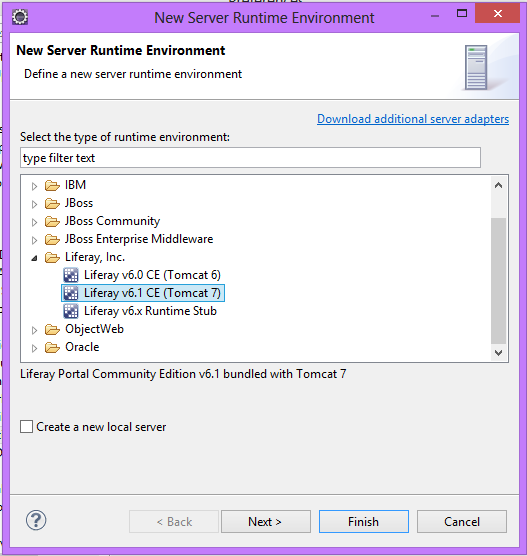
Hình ‑ Liferay SDK configuration 3

Bước 12: Chọn “Server🡪Rumtime Enviroments” ở bên trái và click button “Add” ở bên phải.



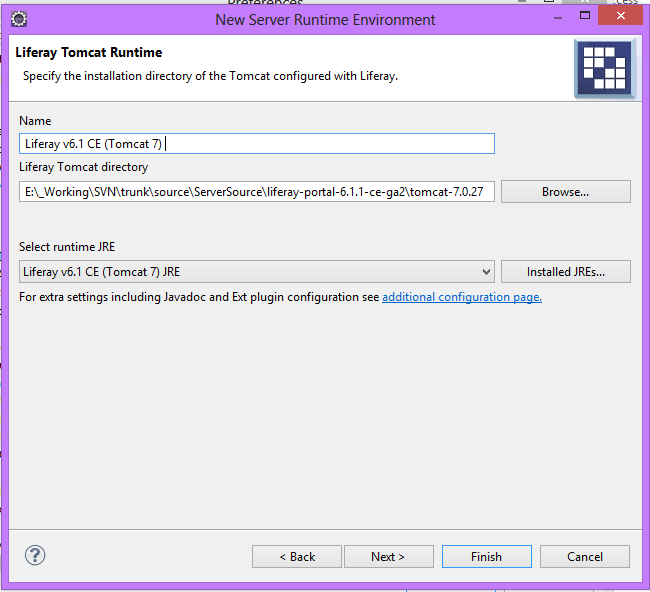
Hình ‑ Configure Liferay Server 1

Bước 13: Chọn thư mục “Liferay v7.0 (Tomcat 7)” trong cửa sổ hiện ra, chọn checkbox “Create a new local server” và click “Next”.



Hình ‑ Configure Liferay Server 2

Bước 14: Trong textbox “Liferay tomcat directory” browse đến thư mục giải nén Liferay tomcat bundle đã download.

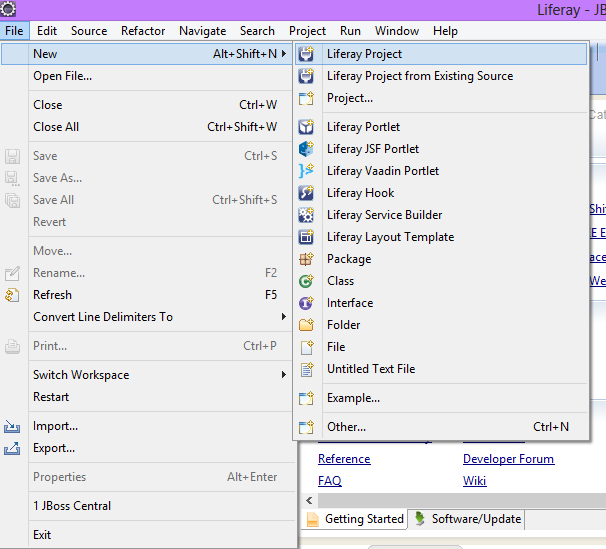


Hình ‑ Configure Liferay Server 3

Bước 15: Chọn “Finish” và chọn “OK” để kết thúc.

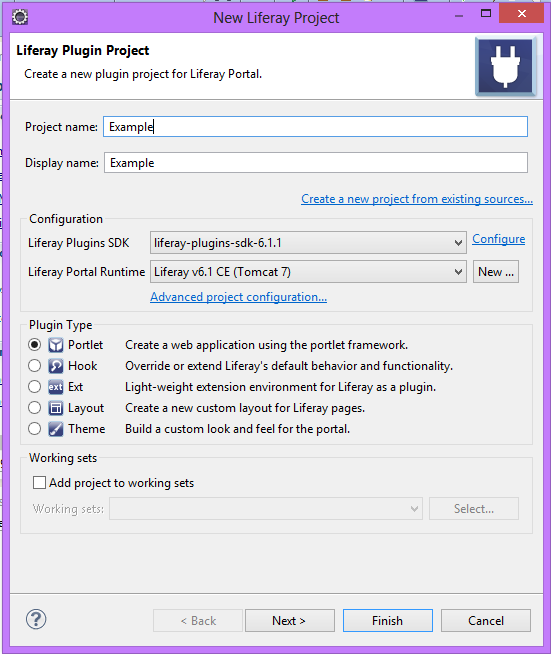
## Tạo portlet trên môi trường Liferay plugins SDK

* Bước 1: Khởi động eclipse. Trong eclipse chọn menu “File🡪New🡪Project…”
* Bước 2: Chọn “Liferay🡪Liferay Project” và click “Next”.



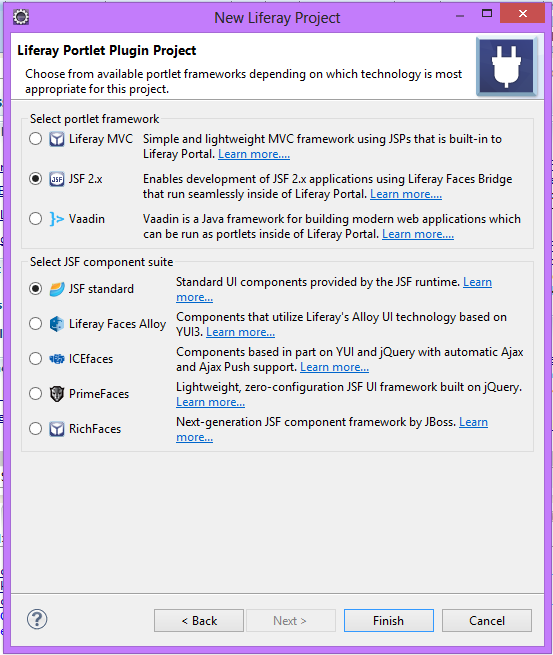
Hình ‑ Tạo mới Liferay Project

Bước 3: Nhập project name rồi click “Next”.



Hình ‑ Nhập thông tin cho Liferay Project

* Bước 4: Chọn loại portlet cần tạo. Liferay plugin SDK hổ trợ bạn tạo 3 loại portlet là “Liferay MVC”, “JSF 2.0” và “Vaadin”.



Hình ‑ Chọn Kiểu của Project

* Bước 5: Chọn “Finish” để hoàn tất việc tạo portlet.

## Cấu hình MySQL trong Liferay

Liferay có một hệ cơ sở dữ liệu mặc định tên là HSQL hay còn gọi là "hypersonic.". Nhưng như thế không có nghĩa là chúng ta phải sử dụng nó. Chúng ta có thể chuyển đổi sang hệ cơ sở dữ liệu mà chúng ta muốn để sử dụng trong Liferay như MySQL, PostgreSQL, Oracle, DB2…Phần sau đây sẽ hướng dẫn chúng ta cấu hình hệ cơ sở dữ liệu MySQL sử dụng cho Liferay.

1. Tạo tập tin **portal-ext.properties.**

Mọi cấu hình về hệ cơ sở dữ liệu sẽ được bỏ trong tập tin **portal-ext.properties**. Bạn có thể tìm thấy tập tin này ở địa chỉ :

**${CATALINA\_HOME}/webapps/ROOT/WEB-INF/classes**

Nếu nó không tồn tại thì bạn sẽ tạo mới nó ra.

1. Cấu hình cho tập tin **portal-ext.properties**

Từ phiên bản 5.2 trở về sau thì mọi cấu hình về cơ sở dữ liệu sẻ bỏ trong tập tin portal**-ext.properties** cho tất cả các ứng dụng của server. Dưới đây là cấu hình cho hệ quản trị cơ sở dữ liệu MySQL:

jdbc.default.driverClassName=com.mysql.jdbc.Driver

jdbc.default.url=jdbc:mysql://localhost/{DatabaseName}?useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false

jdbc.default.username={Username}

jdbc.default.password={Password}

* Với {DatabaseName} là tên của cơ sở dữ liệu
* Với {Username} là tài khoản của MySQL
* Với {Password} là mật khẩu của MySQL

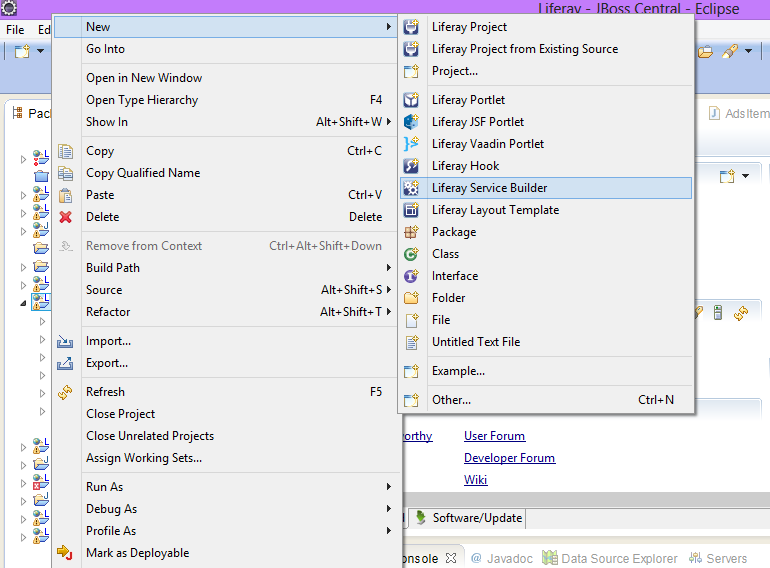
## Service Builder công cụ làm việc với CSDL

### Giới thiệu công cụ service builder

Service builder là một công cụ được xây dựng bởi liferay để tự động hóa việc tạo ra các Interface và Class được sử dụng cho portal hay portlet. Bao gồm code của EJBs, Spring, Persistence, and Model. Service Builder sẽ tạo ra hầu hết các đoạn mã cần thiết để hiện thực các phương thức tìm kiếm tạo đối tượng, cập nhật hay là xóa đối tượng bên trong cơ sở dữ liệu.

Để tạo service builder cho portlet ta thực hiện các bước sau:

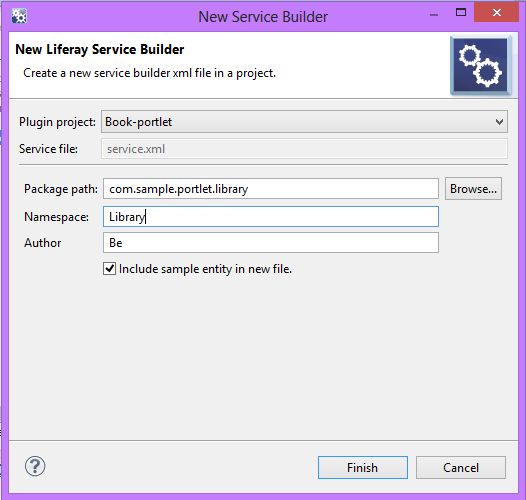
1. Bước 1: Chọn File 🡪 New 🡪 Liferay Service Builder.



Hình ‑ Giao diện chọn Liferay Service Builder

Hình ‑ Giao diện tạo Service Builder

1. Bước 2: Nhập đầy đủ thông tin để tạo service.

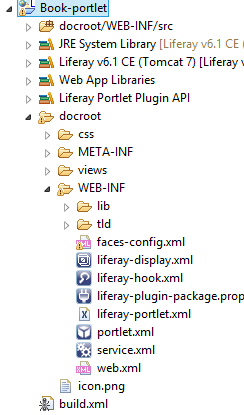


Hình ‑Nhập thông tin

* Plugin project: tên portlet để tạo service.
* Service file: tên file service builder mặc định là service.xml
* Package path: tên package chứa tất cả class và interface trong quá trình tạo build service.
* Namespace: tên của element, để phân biệt 2 nguồn dữ liệu khác nhau trong cùng một file xml
* Author: tên người tạo ra service.

1. Bước 3: Chọn Finish để tạo service.

Cấu trúc portlet sau khi tạo service builder.



Hình ‑ Cấu trúc Service Builder trong porlet

### Khai báo các đối tượng và ánh xạ chúng xuống cơ sở dữ liệu

Việc tạo ra các đối tương và ánh xạ chúng xuống dưới cơ sở dữ liệu được định nghĩa bởi các thẻ XML hỗ trợ sẵn bởi Liferay trong tập tin service.xml.

1. **Package, namespace, author**

Tên của package, author và namespace đã được tạo ra trong bước tạo Service Builder , bạn có thể thay đổi nó bằng cách thay đổi chúng :

<service-builder package-path=*"com.sample.portlet.library"*>

<author>Be</author>

<namespace>Library</namespace>

* Package-path: định nghĩa ra một đường của package, nơi mà các class sẽ được tạo ra.
* Namespace: phải là duy nhất, nó sẽ là phần đầu của tên bảng nếu không định nghĩa tên bảng cho Entity

1. **Khai báo các đối tượng (Entity)**

Các đối tượng và các thuộc tính có liên quan tới đối tượng được khai báo bởi cặp thẻ <entity></entity>

<entity name=*"Book"* local-service=*"true"* remote-service=*"true"*>

* Name : tên của đối tượng cũng chính bảng dưới CSDL
* Local-service: dịch vụ được sử dụng ở local bỏ qua bước kiểm tra permission
* Remote-service: dịch vụ được truy xuất từ xa phải thông qua permission.

1. **Thuộc tính của đối tượng**

Để khai báo các thuộc tính của đối tượng cũng chính là các column dưới cơ sở dữ liệu ta dùng thẻ

<column name=*"bookId"* type=*"long"* primary=*"true"* />

<column name=*"title"* type=*"String"* />

* Primary : khai báo khóa chính cho table (primary=”true”)
* Name : tên đối tượng cũng như tên cột trong database
* Type : kiểu dữ liệu

1. **Thẻ Order**

Service Builder hỗ trợ thẻ order được dùng để sắp xếp giá trị của cột theo thứ tự tăng dần hoặc giảm dần.

<order by=*"asc"*>

<order-column name=*"title"* />

</order>

* By : sắp xếp theo tăng dần (asc) hay giảm dần (dsc)
* Name: tên cột cần sắp xếp

1. **Thẻ Finder**

Service Builder hỗ trợ tìm kiếm dữ liệu thoe cột trong table dưới database bằng cách:

<finder name=*"Title"* return-type=*"Collection"*>

<finder-column name=*"title"* />

</finder>

* Name: tên cột cần tìm kiếm
* Return-type: kểu dữ liệu sẽ được trả về. Nếu trả về một danh sách các đối tường thì ta dùng “Collection” nếu chỉ trả về duy nhất một đối tượng thì ta sử dụng truyền vào trực tiếp tên của đối tượng đó.

1. **Quan hệ giữa các entity**

<entity name=*"Publisher"* local-service=*"true"* remote-service=*"true"*>

<column name=*"publisherId"* type=*"long"* primary=*"true"* />

Primary key of Publisher

<column name=*"publisherName"* type=*"String"* primary=*"true"* />

<column name=*"books"* type=*"Collection"* entity=*"Book"*

mapping-key=*"publisherId"* />

Foreign key relationship

</entity>

<entity name=*"Book"* local-service=*"true"* remote-service=*"true"*>

<column name=*"bookId"* type=*"long"* primary=*"true"* />

<column name=*"title"* type=*"String"* />

<column name=*"publisherId"* type=*""*></column>

<order by=*"asc"*>

Foreign key for Publisher

<order-column name=*"title"* />

</order>

<finder name=*"Title"* return-type=*"Collection"*>

<finder-column name=*"title"* />

</finder>

</entity>

Đoạn mã trên thể hiện quan hệ giữa đối tượng một nhiều giữa đối tượng Publisher và đối tượng Book (one-many relationship). Sau khi build service với đoạn mã trên ta sẽ được 2 bảng Book và Publisher đồng thời Service Builder sẻ sinh ra một phương thức getBooks(long IdPublisher) để lấy được danh sách các book của một publicsher tương ứng.

Tập tin service.xml ví dụ cấu hình hoàn chỉnh

### Ví dụ mẫu file service.xml hoàn chỉnh

<service-builder package-path=*"com"*>

<author>Be</author>

<namespace>Library</namespace>

<entity name=*"Publisher"* local-service=*"true"* remote-service=*"true"*>

<column name=*"publisherId"* type=*"long"* primary=*"true"* />

<column name=*"publisherName"* type=*"String"* />

<column name=*"books"* type=*"Collection"* entity=*"Book"*

mapping-key=*"publisherId"* />

</entity>

<entity name=*"Book"* local-service=*"true"* remote-service=*"true"*>

<column name=*"bookId"* type=*"long"* primary=*"true"* />

<!-- Other fields -->

<column name=*"title"* type=*"String"* />

<column name=*"publisherId"* type=*"long"*></column>

<order by=*"asc"*>

<order-column name=*"title"* />

</order>

<finder name=*"Title"* return-type=*"Collection"*>

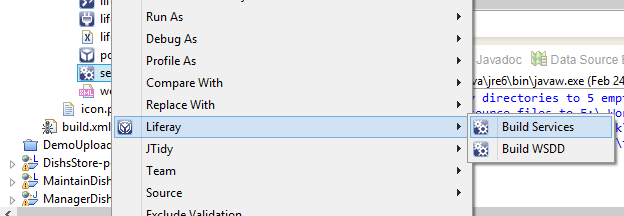
<finder-column name=*"title"* />

</finder>

</entity>

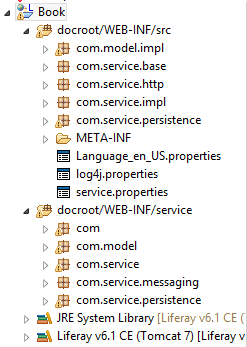
</service-builder>

Sau khi tạo file cấu hình service.xml như trên. Tiến hành build project bằng cách : Chọn build/Liferay/Build Service



Hình ‑ Build Service

Sauk hi Build service thành công sẽ xuất hiện hai package như thế này:



### Xây dựng các phương thức cần thiết trong class {Entity}LocalServiceImpl

Với cấu hình service.xml như trên thì Service Builder sẽ sinh ra hai lớp BookLocalServiceImpl và PublisherLocalServiceImpl. Tại đây chũng ta có thể viết mã của chúng ta mà sau khi build service lại thì sẽ không bị xóa mất.

Sau đây là các đoạn code mình họa việc xây dựng các phương thức cần thiết và phổ biến trong lớp BookLocalServiceImpl :

**public** **class** BookLocalServiceImpl **extends** BookLocalServiceBaseImpl {

**public** Book createBook(Book newBook) **throws** SystemException {

Book book = BookUtil.*create*(CounterLocalServiceUtil

.*increment*(Book.**class**.getName()));

book.setTitle(newBook.getTitle());

book.setAuthor(newBook.getAuthor());

BookUtil.*update*(book, **false**);

**return** book;

}

**public** Book updateBook(Book newBook) **throws** SystemException {

Book book = BookUtil.*fetchByPrimaryKey*(newBook.getBookId());

book.setTitle(newBook.getTitle());

book.setAuthor(newBook.getAuthor());

BookUtil.*update*(book, **false**);

**return** book;

}

**public** Book deleteBook(**long** bookId) **throws** SystemException,

NoSuchBookException {

Book book = BookUtil.*remove*(bookId);

**return** book;

}

**public** Book getBook(**long** bookId) **throws** SystemException {

**return** BookUtil.*fetchByPrimaryKey*(bookId);

}

**public** List<Book> findByBookTitle(String title) **throws** SystemException {

**return** BookUtil.*findByTitle*(title);

}

}

Sau khi đã khai báo các phương thức cần thiết ta tiến hành build lại service builder.

### Tạo class {Entity}Finder để custome thêm các phương thức truy vấn CSDL

Các bước để cấu hình custom-sql và hiện thực

1. Bước 1: Tạo thư mực custom-sql trong **\docroot\WEB-INF\src**
2. Bước 2: Tạo file **default.xml** trong thư mục **custom-sql**, nội dung như sau:

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

<custom-sql>

<sql file=*"custom-sql/book.xml"* />

</custom-sql>

1. Bước 3: Tạo file **book.xml** trong cùng thư mục custom-sql, nội dung như sau:

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

<custom-sql>

<sql id=*"findBooksByName"*>

<![CDATA[

SELECT

\*

FROM

library\_book

WHERE

(library\_book.title like ?)

]]>

</sql>

</custom-sql>

Sau khi khai báo cài cấu hình, cần phải build lại service.

1. Bước 4: Tạo Interface {Entity}Finder

Tạo file **BookFinderImpl.java trong thư mục ..\service\persistence có nội dung như sau:**

**Public class** BookFinderImpl **extends** BasePersistenceImpl<Book>{

**}**

Tiếp theo là cần build lại service builder , sau khi build xong thì nó sẽ tạo ra interface BookFinder. Cập nhật lại **BookFinderImpl.java như sau:**

**Public class** BookFinderImpl **extends** BasePersistenceImpl<Book> implement BookFinder{

**}**

1. Bước 5: cài đặt các phương thức vào lớp **BookFinderImpl.java**

**public** **class** BookFinderImpl **extends** BasePersistenceImpl<Book> **implements**

BookFinder {

**public** **static** String *FIND\_BOOKS* = " *findBooksByName* ";

**public** List<Product> *findBooksByName*s(String bookName)

**throws** SystemException {

Session session = **null**;

**try** {

session = openSession();

String sql = CustomSQLUtil.*get*(*FIND\_ BOOKS*);

SQLQuery query = session.createSQLQuery(sql);

query.addEntity("Book", BookImpl.**class**);

QueryPos qPos = QueryPos.*getInstance*(query);

qPos.add("%"+bookName+"%");

**return** (List) query.list();

} **catch** (Exception e) {}

**return** **null**;

}

}

Build lại service builder.

Bước 6: Cập nhật code trong **BookLocalServiceImpl**.**java** thêm vào phương thức **findBooks(String bookName)**

**public** List<Book> findBooks(String bookName) **throws**

PortalException, SystemException {

**return** BookFinderUtil. *findBooksByName*s (bookName

}

Build lại service builder.

### Link tham khảo :

<http://innovationliferay.blogspot.com/p/service-builder-concept.html>

<http://www.liferay.com/community/wiki/-/wiki/Main/Service+Builder>

<http://www.liferay.com/documentation/liferay-portal/6.0/development/-/ai/service-build-2>

## Web Service

<https://www.liferay.com/documentation/liferay-portal/6.1/development/-/ai/soap-web-services>

Liferay cung cấp dịch vụ try cập thông qua *Simple Object Access Protocol* (*SOAP*) trên HTTP. SOAP là giao thức packaging (*packaging* protocol) và HTTP là giao thức transport (*transport* protocol). Sau đây là cách xây dựng web service dựa trên một ví dụ cụ thể.

### Web service ở Server

Web services ở Server cung cấp các đối tượng và thuộc tính và phương thức của đối tượng đó để Client có thể sử dụng sau khi đã kết nối với server.

Ví dụ của chúng ta gồi hai đối tượng là AdsItem và CategoryAds.Đưới đây là các bước để xây dựng lên web service ở Server

1. Khai báo đối tượng AdsItem và CategoryAds trên công cụ service builder bằng cách khai báo trong file service.xml đoạn mã như sau:

<service-builder package-path=*"irestads"*>

<author>Be</author>

<namespace>Ads</namespace>

<entity name=*"CategoryAds"* local-service=*"true"* remote-service=*"true"*>

<!-- PK fields -->

<column name=*"categoryAdsId"* type=*"long"* primary=*"true"* />

<!-- Other fields -->

<column name=*"categoryAdsName"* type=*"String"* />

<column name=*"adsItemModels"* type=*"Collection"* entity=*"AdsItem"*

mapping-key=*"categoryAdsId"* />

<!-- Order -->

<order by=*"asc"*>

<order-column name=*"categoryAdsName"* />

</order>

<!-- Finder methods -->

<finder name=*"CategoryAdsName"* return-type=*"Collection"*>

<finder-column name=*"categoryAdsName"* />

</finder>

</entity>

<entity name=*"AdsItem"* local-service=*"true"* remote-service=*"true"*>

<!-- PK fields -->

<column name=*"adsItemId"* type=*"long"* primary=*"true"* />

<!-- Other fields -->

<column name=*"companyName"* type=*"String"* />

<column name=*"productName"* type=*"String"* />

<column name=*"numberPhone"* type=*"String"* />

<column name=*"email"* type=*"String"* />

<column name=*"address"* type=*"String"* />

<column name=*"facebook"* type=*"String"* />

<column name=*"twitter"* type=*"String"* />

<column name=*"description"* type=*"String"*></column>

<column name=*"imageContent"* type=*"String"*></column>

<column name=*"timeDuring"* type=*"int"*></column>

<column name=*"tags"* type=*"String"* />

<column name=*"itemType"* type=*"String"* />

<column name=*"categoryAdsId"* type=*"long"*></column>

<!-- Order -->

<order by=*"asc"*>

<order-column name=*"productName"* />

</order>

<!-- Finder methods -->

<finder name=*"ProductName"* return-type=*"Collection"*>

<finder-column name=*"productName"* />

</finder>

<finder name=*"TimeDuring"* return-type=*"Collection"*>

<finder-column name=*"timeDuring"* />

</finder>

</entity>

</service-builder>

1. Build service builder và build Web service

* Build Service Builder bằng cách click phải vào **build.xml** chọn **Liferay** chọn **Build Services**
* Sau khi build service builder thành công thì sẽ service builder sẽ sinh ra các class, interface, and model cần thiết. Chúng ta cần quan tâm tới các class gọi là SOAP để sử dụng trong web service sau:

AdsItemSoap.java

CategorySoap.java

AdsItemServiceSoap.java

CategoryServiceSoap.java

* Các lớp {Entity}ServiceSoap là interface để định danh các service trên tập tin WSDL (*Web Services Definition Language*).Còn các lớp {Entity}Soap là các serializeable hiện thực lại các model.
* Sau khi build service buider thành công thì ta tiến hành build Web service bằng cách Click phải vào **build.xml** chọn **Liferay** chọn **Build WSDD**

1. Hiện thức các phương thức cần thiết trong lớp {Entity}ServiceImpl

**public** AdsItem createAdsItem(String companyName, String numberPhone,

String email, String address, String facebook, String twitter,

String description, String imageContent, **int** timeDuring,

String productName, String tags, String itemType, **long** categoryAdsId) {

// mã hiện thực ở đây

}

**public** AdsItem createAdsItem(AdsItem adsItem) {

// mã hiện thực hiện ở đây

}

**public** AdsItem updateAdsItem(AdsItem item) {

// mã hiện thực hiện ở đây

}

**public** AdsItem deleteAdsItem(**long** adsItem) {

// mã hiện thực hiện ở đây

}

**public** AdsItem findById(**long** adsItemId) {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> getAllAdsItems() {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> findAdsByCompName(String keyword) {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> findAdsByTags(String keyword) {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> findAdsByProductName(String keyword) {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> findAdsByItemType(String keyword) {

// mã hiện thực hiện ở đây

}

**public** List<AdsItem> findAdsByCategoryName(String keyword) {

// mã hiện thực hiện ở đây

}

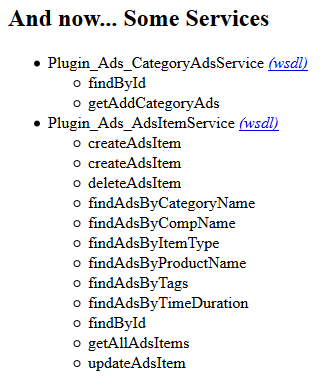
**public** List<AdsItem> findAdsByTimeDuration(**int** keyword) {

// mã hiện thực hiện ở đây

}

1. Xác định URL cho các service

Xem danh sách các services được deploy trên portal bằng cách mở trình duyệt lên và nhập vào URL thoe định dạng <http://[host]:[port]/api/secure/axis> đối với cách service có chết độ bảo mật và URL theo định dạng <http://[host]:[port]/api/axis> đối với các service không yêu cầu chứng thực của người dùng. Nhập vào link [http://[host]:[port]/[porlet name]-porlet/api/axis](http://[host]:[port]/%5bporlet%20name%5d-porlet/api/axis) để truy cập trực tiếp vào một cái web services cụ thể. Dươi đây là web service cho ví dụ của chúng ta:



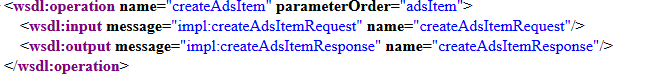
Hình ‑ CategoryAds Web Service và AdsItem Web Service

Các phương thức trong web service được hiện thực trong lớp {Entity}ServiceImpl.

Với từng web service được liệt kê với tên, các phương thức và link tới tập tin WSDL của nó.

1. Chi tiết web service

Click vào link wsdl của từng web servie để xem chi tiết về web service đó .



Hình ‑ Phương thức createAdsItem của đối tượng AdsItem được hiện thực trên web service

# NỘI DUNG NGHIÊN CỨU: PRIME FACE

## Giới thiệu về JSF

### Khái niệm về JSF

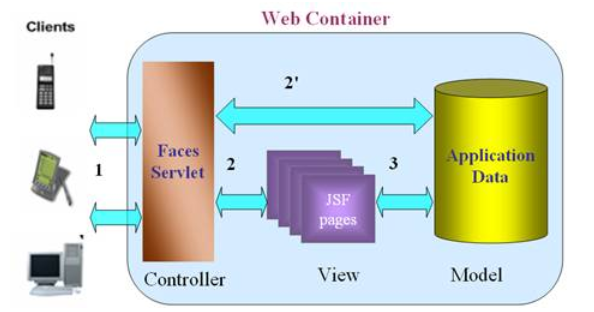
* **JSF** là viết tắt của **JavaServer Faces,** JSF là một [bộ khung](http://vi.wikipedia.org/w/index.php?title=B%E1%BB%99_khung&action=edit&redlink=1) (framework) phát triển các ứng dụng Web viết bằng Java nhằm làm đơn giản hóa quá trình [phát triển](http://vi.wikipedia.org/wiki/C%C3%B4ng_ngh%E1%BB%87_ph%E1%BA%A7n_m%E1%BB%81m) [giao diện người dùng](http://vi.wikipedia.org/w/index.php?title=Giao_di%E1%BB%87n_ng%C6%B0%E1%BB%9Di_d%C3%B9ng&action=edit&redlink=1) cho các ứng dụng [J2EE](http://vi.wikipedia.org/wiki/J2EE). Để tạo ra giao diện hiển thị, JSF dùng dạng cấu trúc cây của các thẻ, mỗi thẻ là một thành phần giao diện (*component*) và **FacesServlet** servlet sẽ thực hiện công đoạn chuyển đổi ra giao diện tương ứng cho người dùng với định dạng HTML.
* JSF Framework được xây dựng trên nền tảng MVC Pattern trên Web và hỗ trợ khả năng sử lý sự kiện điều này tương tự như mô hình Swing – xây dựng ứng dụng desktop. Do vậy, JSF còn được mệnh danh là Swing được chạy trên phía Server (Server Side).
* JSF còn hỗ trợ cơ chế rendering trong việc hỗ trợ giao diện cho người dùng trong quá trình xử lý (User Interface) và đây là đặc tính nổi bật của JSF theo nghĩa JSF đa số hỗ trợ chữ V (View) trong mô hình MVC chuẩn (event – biến cố là một phần trong View)
* Do tính chất hỗ trợ giao diện cho nên JSF đưa ra một tập các thư viện trên trang jsp – tag library để hỗ trợ giao diện, checking ràng buộc – validation, ép kiểu dữ liệu – conversion, hỗ trợ đa ngôn ngữ - localization, …

### Mục tiêu của JSF

* Tạo ra một bộ khung gồm các thành phần giao diện người dùng chuẩn (*standard GUI component framework*) nhằm giúp cho các công cụ phát triển dễ dàng hơn cho người dùng trong việc tạo GUI chất lượng cao đồng thời quản lí các kết quả của GUI với xử lí thực thi của chương trình.
* Cung cấp một mô hình JavaBeans để có thể truyền đi (dispatch) các sự kiện từ các GUI controls phía máy khách đến các xử lí hiện thực cụ thể từ phía ứng dụng máy chủ.
* Định ra các hàm APIs để kiểm chứng dữ liệu nhập, bao gồm hỗ trợ kiểm chứng từ phía máy chủ.

## Các thành phần của JSF Framework

### Kiến trúc của JSF



Hình ‑ Mô hình MVC

* Mô hình MVC thể hiện rất rõ ràng, khi người dùng request, toàn bộ request bắt buộc phải qua Controller. Ở dây JSF Framework đặt tên Servlet là Faces và che nó lại và cho phép người lập trình mapping linh động không bị code lệ thuộc như bình thường trong mô hình MVC thông qua tập tin faces-config.xml, nơi người dùng khái báo các managed beans, mapping việc chọn lựa trang xử lý, chuyển xử lý và truyền thông số giữa các bean và các trang
* Sau khi tiếp nhận, Faces Servlet xác định bean tương ứng – Model để chuyển xử lý hay xác định trang JSF – View tương tự như cơ chế MVC xác định trang JSP và Model tương ứng để thực hiện xử lý và kết xuất dữ liệu lên form JSF để trình bày
* Kết quả, xử lý hay giao diện kết xuất phải được đưa qua Faces Servlet để chuyển response về cho người yêu cầu

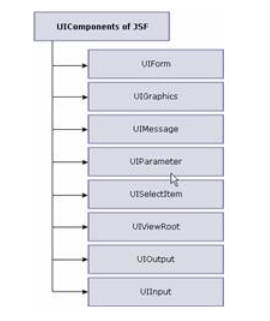
### Managed Bean

Managed Bean hay tên gọi khác là Backing Bean

* Bản chất là Java Beans được dùng kết hợp với các thành phần UI interface, Java Beans bao gồm các phương thức hỗ trợ trình bày giao diện cho người dùng, trao đổi dữ liệu với các thành phần khác trong ứng dụng, lưu trữ và chuyển xử lý khi người dùng tương tác
* Hỗ trợ 5 scope khác nhau trong ứng dụng khi thao tác như request, session, view, none và application
* Java beans phải được khai báo trong tập tin faces-config.xml của ứng dụng để ứng dụng có thể cơ chế để lưu trữ và tương tác với bean
* Để tương tác giữa form, UI component thì Java Beans sử dụng cấu trúc #{[JavaBeanName].[property/method]} với [JavaBeanName]là tên java beans khi ta khai báo lớp bin và [property/method] là tên thuộc tính hoặc phương thức của lớp java beans
* Beans có khả năng hỗ trợ Event Handling, hỗ trợ chuyển trang – Navigation, và thực hiện Validation

### UI Component

Hỗ trợ trình bày giao diện để người dùng tương tác trực tiếp và trình bày kết quả về phía người dùng



Hình ‑ JSF component

### Renderer

* Hỗ trợ tính toán, xử lý kết quả để kết xuất trình bày giá trị trong các thành phần UI để trình bày giao diện chính xác và hợp lý từ kết xuất của chương trình trong lúc thực thi.
* Hỗ trợ cơ chế “Look and Feel”.

### Events và Listeners

Hỗ trợ cơ chế đón nhận các sự kiện và xử lý sự kiện (Event Handling) trên khái niệm của M (Model) trong MVC kết hợp với Managed Bean

### Navigation

Cung cấp cơ chế chuyển trang, chuyển xử lý trong quá trình ứng dụng thực hiện tượng tự khái niệm C (Controller) trong mô hình MVC

Có 02 loại mô hình Static và Dynamic Model

* Static Model: di chuyển đến duy nhất một trang cụ thể và không thay đổi hay không thể thay đổi từ lúc lập trình hay trong quá trình trang thực thi (trang chuyển đến đã được xác định chính xác)
* Dynamic Model: cấu hình để chuyển đến trang tương ứng tùy theo kết quả xử lý của ứng dụng

### Validator

Hỗ trợ cơ chế kiểm tra dữ liệu trên các thành phần UI để đảm bảo dữ liệu được nhập đúng và có ràng buộc theo đúng yêu cầu của ứng dụng

### Converter

Hỗ trợ cơ chế ép kiểu dữ liệu trong quá trình xử lý

### Message

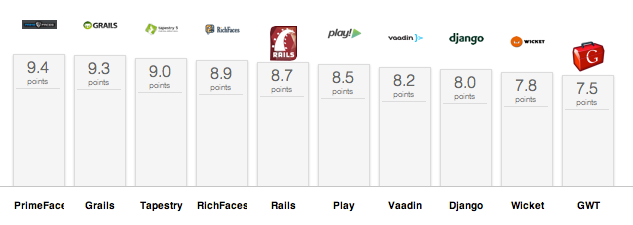
Hỗ trợ trình bày các thông tin – message về phía người dùng, thông báo lỗi hay hỗ trợ cơ chế I18N – hỗ trợ ứng dụng đa ngôn ngữ

## Giới thiệu Primeface

PrimeFace framework được mở rộng từ JSF. Đa dạng về các thành phần (HtmlEditor, Dialog, AutoComplete, Charts và nhiều thành phần khác).Xây dựng Ajax dựa trên chuẩn JSF 2.0 Nhẹ, không có cấu hình phức tạp.

**Đánh giá về primeface**

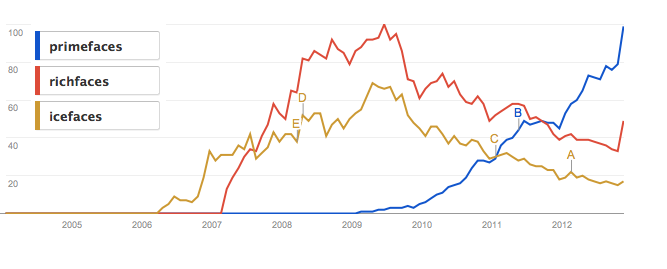
Theo [DevRates.com](http://devrates.com/stats/index) thì PrimeFace được đánh giá với đển 9.4 số với các framework khác



Hình ‑ Kết quả đánh giá về primeface của [DevRates.com](http://devrates.com/stats/index)

Kết quả trên cho ta thấy frimeface được đánh giá rất tốt, được tin cậy và sử dụng khá nhiều.

**So sánh primeface vơi các framework tương tự**

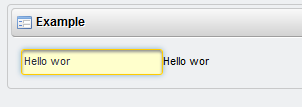
****

Hình ‑ Kết quả so sánh của PrimeFace

Hàng ngàn ứng dụng đã được tạo ra với PrimeFaces, sản phẩm thậm chí cạnh tranh như IceFaces được cung bởi PrimeFaces. Trên đây là một biểu đồ được tạo ra bởi Google Trends so sánh các PrimeFaces phổ biến với các đối thủ cạnh tranh.

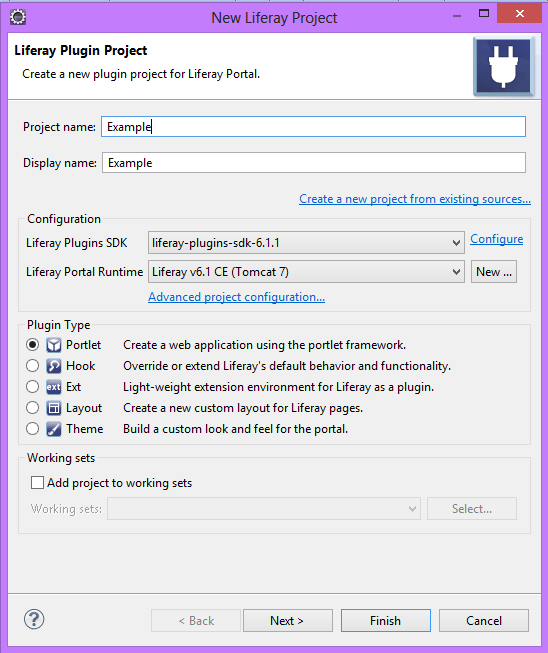
## Tạo Project đơn giản với liferay portal và primeface

Mô tả ví dụ: giao diện gồm một input text khi người dùng nhập chữ tới đâu thì bên cạnh input text đó sẽ xuất hiện những chữ và người dùng vừa nhập.



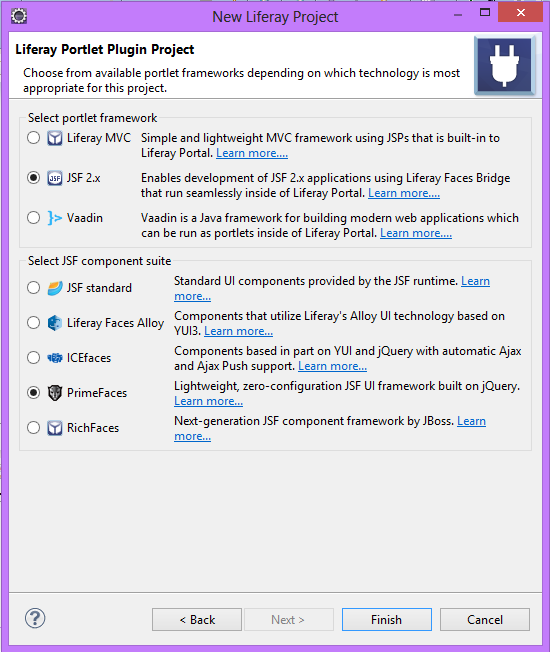
Kết quả của ví dụ

* + - 1. Bước 1: Tạo mới một porlet



Tạo mới project

* + - 1. Bước 2: Click và Chọn JSF và PrimeFace

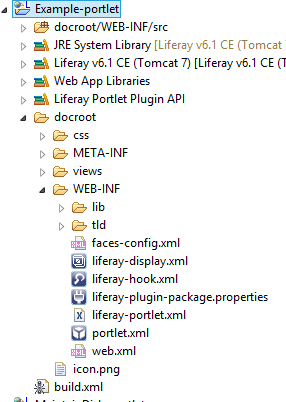


Chọn JSF và Primeface

Tại bước này chọn JSF và chọn PrimeFace

* + - 1. Bước 3: chọn finish

Cấu trúc của project khi được tạo, chú ý rằng khi chọn JSF liferay portal sẽ tự động sinh ra tập tin face-config.xml



* + - 1. Bước 4: Tạo lớp java bean
         1. Tạo lớp java tên ExampleBean với thuộc tính userName với nội dung sau :

**public** **class** ExampleBean {

**private** String userName;

**public** ExampleBean() {

// **TODO** Auto-generated constructor stub

}

**public** String getUserName() {

System.*out*.println("name "+**this**.userName);

**return** userName;

}

**public** **void** setUserName(String userName) {

System.*out*.println("set un "+**this**.userName);

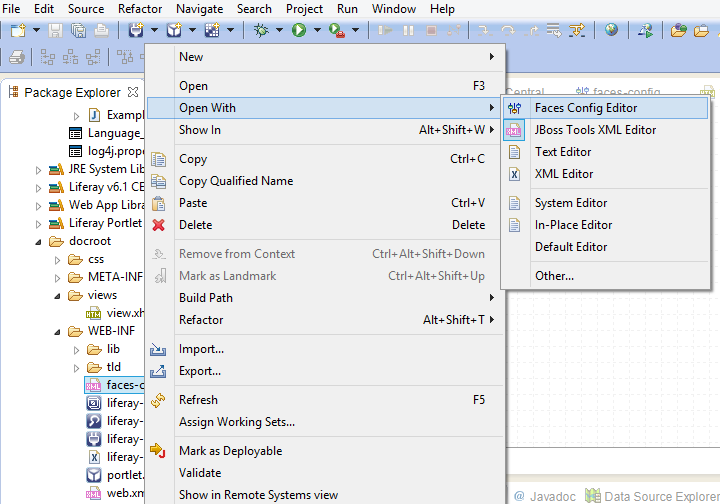
**this**.userName = userName;

}

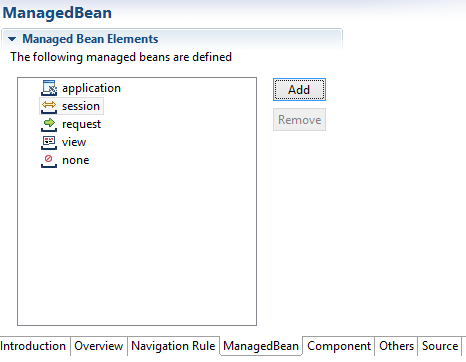
* + - * 1. }

Chú ý khi khai báo thuộc tính phải tạo phương thức get và set cho thuộc tính đó nếu không thì lớp bean sẽ không gọi được thuộc tính đó trên view và cũng không gán giá trị cho thuộc tính đó được. Đồng thời phải tạo constrer cho lớp đó để khởi tạo đối tượng bean trên server.

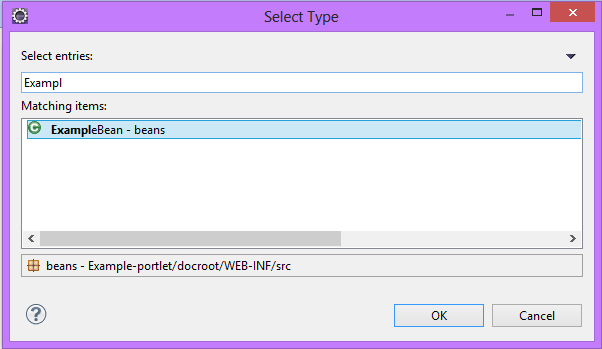
* + - * 1. Mởi file **face-config.xml** bằng công cụ **face config editer** để khai báo lớp bean bằng cách sau: click chuột phải vào **face-config.xml** và chọn **open with** và **chọn face config editer**



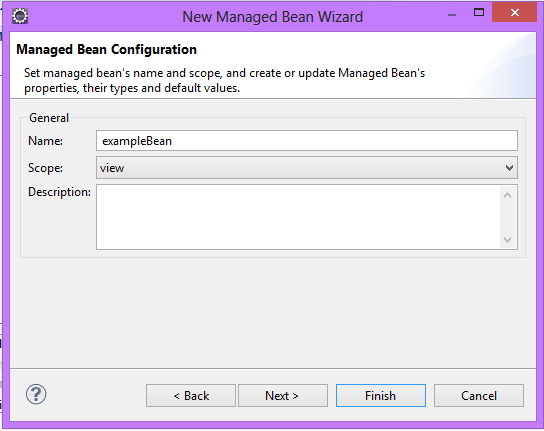
* + - * 1. Chọn tag Managed bean



* + - * 1. Chọn Add sau đó chọn Browse sẽ hiển thị ra cửa sổ để chọn lớp Java để sử dụng làm lớp bean. Tiếp thoe là điền vào tên của lớp java mà muốn khai báo là lớp bean, cụ thể ở ở ví dụ này nó có tên giống hình dưới. Sau khi đã tìm được lớp java thì click vào nó và nhấn OK



* + - * 1. Chọn Next
        2. Chọn scope sau đó chọn finish



Name: tên của lớp bean để gọi trên view

Sau khi chọn finish thì việc khai báo lớp bean của chúng đã thành công và nó sẽ được gọi trên view với tên exampleBean.

* + - 1. Bước 5 nhấp dôi vào thư mục views và chọn file view.xml hoặc có thể tạo file xml mới và sau đó khai báo đoạn mã ví dụ sau:

<f:view xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:c=*"http://java.sun.com/jsp/jstl/core"*

xmlns:f=*"http://java.sun.com/jsf/core"*

xmlns:h=*"http://java.sun.com/jsf/html"*

xmlns:p=*"http://primefaces.org/ui"*

xmlns:ui=*"http://java.sun.com/jsf/facelets"*>

<h:head />

<h:body>

<h:form>

<p:inputText value=*"#{exampleBean.userName}"*>

<p:ajax event=*"keyup"* update=*"nameLable"*></p:ajax>

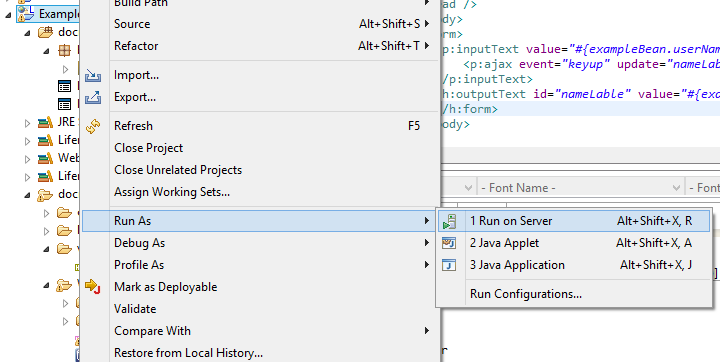
</p:inputText>

<h:outputText id=*"nameLable"* value=*"#{exampleBean.userName}"* />

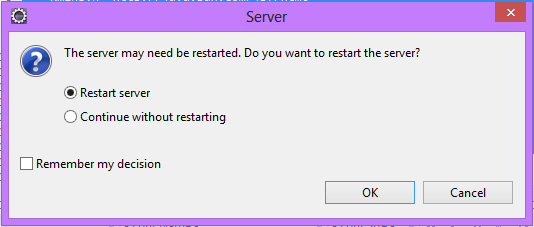
</h:form>

</h:body>

* + - 1. Bước 6: Tiến hành chạy ứng dụng với server
         1. Click chuột phải vào project chọn Run As sau đó chọn tiếp Run on Server như hình dưới:

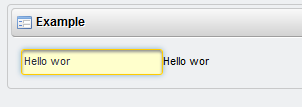


* + - * 1. Chọn Finish xẽ xuốt hiện cửa sổ



Chọn ok nếu server của bạn chưa restar và chọn cái còn lại nếu server của bạn đã restart thành công.

* + - 1. Hiển thị kết quả



Sau khi chạy ứng dụng với server thành công thì sẽ hiển thị kết quả như hình trên. Khi người dùng vừa nhập chữ vào input text thì sẽ hiển thị ngay lập tức chữ người dùng vừa nhập.

## PrimeFace Component

Một số tag phổ biến của PrimeFace

|  |  |
| --- | --- |
| Tag | Mô tả |
| [cellEditor](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/cellEditor.html) | CellEditor được dùng trong datatable sử dụng cho việc sửa đổi giá trị trong một ô của table. |
| [**commandButton**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/commandButton.html) | CommandButton mở rộng thêm từ JSF cơ bản và có thêm các tính năng về ajax và giao diện. |
| [**commandLink**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/commandLink.html) | CommandLink mở rộng thêm từ commandLink của JSF cơ bản và thêm một số tính năng về ajax. |
| [**confirmDialog**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/confirmDialog.html) | ConfirmDialog thay thế cho hộp thoại trong Javascript, hỗ trợ giao diện, tùy biến mạnh. |
| [**contextMenu**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/contextMenu.html) | ContextMen hỗ trợ hiện thị menu khi right click |
| [**dataGrid**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/dataGrid.html) | DataGrid hỗ trợ hiển thị danh sách dữ liệu theo bố cục dạng lưới(grid layout).Hỗ trợ phân trang với tùy chọn mẫu phân trang, số dọng trên một trang và nhiều tùy chọn hơn nữa. |
| [**dataTable**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/dataTable.html) | DataTable hỗ trợ hiển thị danh sách dữ liệu theo cột và nhiều chức năng hỗ trợ như phân trang, sắp xếp, chọn, lazy loading, lọc dữ liệu và nhiều hơn nữa. |
| [**fileUpload**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/fileUpload.html) | FileUpload cho phép trỏ tới file cần chọn đồng thời hỗ trợ các tính năng của flash-javascript cho việc upload nhiều file.Hỗ trợ lọc file, upload nhiều file,cập nhật một phần trang, thanh tiến trình. |
| [**graphicImage**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/graphicImage.html) | GraphicImage mở rộng JSF graphic image với khả năng hiện thị dữ liệu theo kiểu binary giống như là một inputstream.Hỗ trợ hiển thị hình ảnh từ database dễ dàng hơn. |
| [**growl**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/growl.html) | Growl sử dụng để hiển thị FacesMessages giống như là h:messages |
| [**inputMask**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/inputMask.html) | InputMask định nghĩa các khuân mẫu cho dữ liệu nhập vào như: email, số điện thoại…. |
| [**inputText**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/inputText.html) | InputText mở rộng inputText của JSF cơ bản với hỗ trợ về giao diện. |
| [**inputTextarea**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/inputTextarea.html) | InputTextarea mở rộng inputTextarea của JSF cơ bản với hỗ trợ về giao diện và khả năng tự động tăng kích thước khi tràn nội dung. |
| [**message**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/message.html) | Hiển thị thông báo khi có lỗi xảy ra tại phần tử xác định. |
| [**messages**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/messages.html) | Hiển thị tất cả các thông báo khi có lỗi xảy ra. |
| [**panelGrid**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/panelGrid.html) | Hiểu thì dữ liệu thoe cột bằng cách định nghĩa số lượng cột |
| [**rowEditor**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/rowEditor.html) | Được khai báo trong datatable dùng để sửa đổi dữ liệu của một dòng trong table. |
| [**selectOneMenu**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/selectOneMenu.html) | Giống như Dropdown ở HTML |
| **selectOneRadio** | Giống như button ở HTML |
| [**subTable**](http://www.primefaces.org/docs/vdl/3.5/primefaces-p/subTable.html) | Được khai báo trong datatable dùng để gom nhóm dữ liệu trong table |

**Attribute**

| **Name** | **Required** | **Type** | **Description** |
| --- | --- | --- | --- |
| id | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Định danh sự duy nhất của phần tử. |
| rendered | false | javax.el.ValueExpression (*must evaluate to* java.lang.Boolean) | Rendered=true hiển thị phần tử, rendered=false ko hiển thì phần tử. |
| value | false | javax.el.ValueExpression (*must evaluate to* java.lang.Object) | Giá trị của phần tử. |
| actionListener | false | javax.el.ValueExpression (*must evaluate to* javax.faces.event.ActionListener) | Một actionlistener được gọi thực hiện khi mà một lệnh nòa đó đã được thực thi. |
| action | false | javax.el.ValueExpression (*must evaluate to* javax.el.MethodExpression) | Một phương thức hoặckếtquảmột chuỗiđể xử lýkhi lệnhđược thực thi. |
| ajax | false | javax.el.ValueExpression (*must evaluate to* java.lang.Boolean) | Ajax=true(mặc định),thì submit sẽ được thực hiện với Ajax. |
| async | false | javax.el.ValueExpression (*must evaluate to* java.lang.Boolean) | Khi giá trị này là true, thì yêu cầu ajax sẽ thực hiện bất đồng bộ. |
| process | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Component sẽ sử lý cục bộ một phần thay vì toàn bộ giao diện. |
| update | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Dùng để khai báo các component được cập nhật từ ajax. |
| onstart | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Trước khi yêu cầu ajax được thực thi nó sẽ được phản hồi ở phía người dùng. |
| oncomplete | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi yêu cầu ajax hoàn tất sẽ có phản hồi ở phía người dùng. |
| onerror | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi yêu cầu ajax thất bại sẽ có phản hồi ở phía người dùng. |
| onsuccess | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi yêu cầu ajax gửi thành công sẽ có phản hồi ở phía người dùng. |
| disabled | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Vô hiệu hoá nút(button). |
| label | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Đặt tên cho thành phần  . |
| tabindex | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Vị trí của thành phần nút theo thứ tự tab. |
| title | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Thông tin gợi ý. |
| type | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Thiết lập hành vi cho nút. Mặc định là submit. |
| readonly | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Chỉ cho đọc |
| style | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Kiểu inline cho các thành phần nút. |
| styleClass | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Lớp kiểu cho nút. |
| onblur | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Sau khi rời khỏi nút sẽ có một phản hồi ở phía người dùng. |
| onchange | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi rời khỏi nút và giá trị đã thay đổi sẽ có phản hồi ở phía người dùng. |
| onclick | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhấn nút sẽ có phản hổi từ ở phía người dùng. |
| ondblclick | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhấn đúp chuột vào nút sẽ có phản hồi ở phía người dùng. |
| onfocus | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi focus vào nút sẽ có một phản hồi ở phía người dùng. |
| onkeydown | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi ấn giữ phím sẽ có phản hồi ở phía người dùng. |
| onkeypress | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhấn rồi nhả phím sẽ có phản hồi ở phía người dùng. |
| onkeyup | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhả phím sẽ có phản hồi ở phía người dùng. |
| onmousedown | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhấn giữ chuột lên nút sẽ có phản hồi ở phía người dùng. |
| onmousemove | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi di trỏ chuột sẽ có phản hồi ở phía người dùng. |
| onmouseout | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi di chuột ra khỏi nút sẽ có phản hồi ở phía người dùng. |
| onmouseover | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi di chuột lên nút sẽ có phản hồi ở phía người dùng. |
| onmouseup | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Khi nhả chuột sau khi đã nhấn lên nút sẽ có phản hồi ở phía người dùng. |
| icon | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Biểu tượng của nút. |
| iconPos | false | javax.el.ValueExpression (*must evaluate to* java.lang.String) | Vị trí của biểu tượng, mặt định là left (bên trái). |
| inline | false | javax.el.ValueExpression (*must evaluate to* java.lang.Boolean) | Hiển thị nút kiểu inline thay vì phù hợp với chiều rộng của nội dung, chỉ sử dụng ở mobile. |

## Hướng dẫn viết các tag prime faces

### Khai báo cơ bản

Khai báo lớp java bean “TableBean” như sau để làm các ví dụ ở các phần tiếp thoe.

**public** **class** TableBean **implements** Serializable {

**private** **final** **static** String[] *colors*;

**private** **final** **static** String[] *manufacturers*;

**private** List<Car> carsSmall;

**static** {

*colors* = **new** String[10];

*colors*[0] = "Black";

*colors*[1] = "White";

*colors*[2] = "Green";

*colors*[3] = "Red";

*colors*[4] = "Blue";

*colors*[5] = "Orange";

*colors*[6] = "Silver";

*colors*[7] = "Yellow";

*colors*[8] = "Brown";

*colors*[9] = "Maroon";

*manufacturers* = **new** String[10];

*manufacturers*[0] = "Mercedes";

*manufacturers*[1] = "BMW";

*manufacturers*[2] = "Volvo";

*manufacturers*[3] = "Audi";

*manufacturers*[4] = "Renault";

*manufacturers*[5] = "Opel";

*manufacturers*[6] = "Volkswagen";

*manufacturers*[7] = "Chrysler";

*manufacturers*[8] = "Ferrari";

*manufacturers*[9] = "Ford";

}

**public** TableBean() {

carsSmall = **new** ArrayList<Car>();

populateRandomCars(carsSmall, 9);

}

**private** **void** populateRandomCars(List<Car> list, **int** size) {

**for**(**int** i = 0 ; i < size ; i++)

list.add(**new** Car(getRandomModel(), getRandomYear(), getRandomManufacturer(), getRandomColor()));

}

**public** List<Car> getCarsSmall() {

**return** carsSmall;

}

**private** **int** getRandomYear() {

**return** (**int**) (Math.*random*() \* 50 + 1960);

}

**private** String getRandomColor() {

**return** *colors*[(**int**) (Math.*random*() \* 10)];

}

**private** String getRandomManufacturer() {

**return** *manufacturers*[(**int**) (Math.*random*() \* 10)];

}

**private** String getRandomModel() {

**return** UUID.randomUUID().toString().substring(0, 8);

}

**public** String[] getManufacturers() {

**return** *manufacturers*;

}

**public** String[] getColors() {

**return** *colors*;

}

**public** **void** onEdit(RowEditEvent event) {

FacesMessage msg = **new** FacesMessage("Car Edited", ((Car) event.getObject()).getModel());

FacesContext.getCurrentInstance().addMessage(**null**, msg);

}

**public** **void** onCancel(RowEditEvent event) {

FacesMessage msg = **new** FacesMessage("Car Cancelled", ((Car) event.getObject()).getModel());

FacesContext.getCurrentInstance().addMessage(**null**, msg);

}

}

}

### Data table

DataTable là một phiên bản nâng cao của DataTable nó cung cấp các giải phát cho nhiều trường hợp sử dụng phổ biến như phân trang, sắp xếp, chọn lựa, lọc và nhiều chức năng khác.



Sử dụng lớp Car và TableBean trong phần khai báo cơ bản.

1. Tạo data table cơ bản

<p:dataTable var=*"car"* value=*"#{carBean.cars}"*>

<p:column>

<f:facet name=*"header"*>

<h:outputText value=*"Model"* />

</f:facet>

<h:outputText value=*"#{car.model}"* />

</p:column>

<p:column>

<f:facet name=*"header"*>

<h:outputText value=*"Year"* />

</f:facet>

<h:outputText value=*"#{car.year}"* />

</p:column>

<p:column>

<f:facet name=*"header"*>

<h:outputText value=*"Manufacturer"* />

</f:facet>

<h:outputText value=*"#{car.manufacturer}"* />

</p:column>

<p:column>

<f:facet name=*"header"*>

<h:outputText value=*"Color"* />

</f:facet>

<h:outputText value=*"#{car.color}"* />

</p:column>

</p:dataTable>

1. Header and Footer

Khai báo header và footer cho table

<p:dataTable var=*"car"* value=*"#{carBean.cars}"*>

<f:facet name=*"header"*>

List of Cars

// Khai báo các cột ở đây

</p:dataTable>

1. Pagination

DataTable đã tích hợp sẵn phân trang dựa trên ajax được kích hoạt bằng cách thiết lập tùy chọn paginator=true.

<p:dataTable var=*"car"* value=*"#{carBean.cars}"* paginator=*"true"*rows=*"10"*>

//columns

</p:dataTable>

1. Sắp xếp

Khai báo datatable với chức năng sắp xếp dữ liệu theo cột bằng cách khai báo thuộc tính sortBy

<p:dataTable var=*"car"* value=*"#{carBean.cars}"*>

<p:column sortBy=*"#{car.model}"* headerText=*”Model”*>

<h:outputText value=*"#{car.model}"* />

</p:column>

...more columns

</p:dataTable>

1. Lọc dữ liệu

Khai báo datatable với chức năng lọc dữ liệu theo cột bằng cách khai báo thuộc tính filterBy

<p:dataTable var=*"car"* value=*"#{carBean.cars}"*

filteredValue=*"#{carBean.filteredCars}"*>

<p:column filterBy=*"#{car.model}"* headerText=*"Model"*>

<h:outputText value=*"#{car.model}"* />

</p:column>

...more columns

</p:dataTable>

1. Row Selection

Row selection dùng để chọn một đối tượng trong data table.

*CarBean.class*

**public** **class** CarBean {

**private** List<Car> cars;

**private** Car selectedCar;

**private** Car[] selectedCars;

**public** CarBean() {

cars = **new** ArrayList<Car>();

cars.add(**new** Car("myModel",2005,"ManufacturerX","blue"));

//add more cars

}

//getters and setters

}

**Single Selection with Row Click**

<p:dataTable var=*"car"* value=*"#{carBean.cars}"* selectionMode=*"single"*

selection=*"#{carBean.selectedCar}"* rowKey=*"#{car.id}"*>

...columns

</p:dataTable>

**Multiple Selection with Row Click**

<p:dataTable var=*"car"* value=*"#{carBean.cars}"* selectionMode=*"multiple"*

selection=*"#{carBean.selectedCars}"* rowKey=*"#{car.id}"* >

...columns

</p:dataTable>

**Single Selection with RadioButton**

<p:dataTable var=*"car"*value=*"#{carBean.cars}"*selection=*"#{carBean.selecedCar}"*

rowKey=*"#{car.id}"*>

<p:column selectionMode=*"single"*/>

...columns

</p:dataTable>

**Multiple Selection with Checkboxes**

<p:dataTable var=*"car"* value=*"#{carBean.cars}"* selection=*"#{carBean.selecedCars}"*

rowKey=*"#{car.id}"* >

<p:column selectionMode=*"multiple"*/>

...columns

</p:dataTable>

1. **Scrolling**

Scolling là cách hiển thị dữ liệu với chiều cao của data table là cố định.

<p:dataTable var=*"car"*value=*"#{bean.data}"*scrollable=*"true"*scrollHeight=*"150"*>

<p:column width=*"100"* />

//columns

</p:dataTable>

1. **Incell Editing**

<p:dataTable var=*"car"* value=*"#{carBean.cars}"*>

<f:facet name=*"header"*>

In-Cell Editing

</f:facet>

<p:column headerText=*"Model"*>

<p:cellEditor>

<f:facet name=*"output"*>

<h:outputText value=*"#{car.model}"* />

</f:facet>

<f:facet name=*"input"*>

<h:inputText value=*"#{car.model}"* />

</f:facet>

</p:cellEditor>

</p:column>

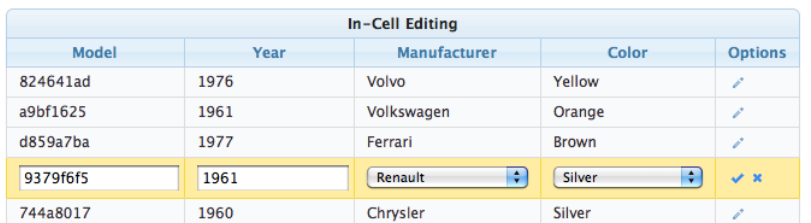
//more columns with cell editors

<p:column>

<p:rowEditor />

</p:column>

</p:dataTable>

****

**Sửa đổi dữ liệu của hàng.**

**Chú ý : Do thời gian có hạn nên không thể trình bày hết tất cả các thẻ của primefce được.Nên đễ xem cách hiện thực của một số component khác thì tham khảo tại trang của primeface :** [**http://www.primefaces.org/showcase/ui/ajaxifyKeyEvents.jsf**](http://www.primefaces.org/showcase/ui/ajaxifyKeyEvents.jsf)