# GIỚI THIỆU ĐỀ TÀI

## HIỆN TRẠNG

### Điện thoại di động ngày càng trở nên gắn bó với hoạt động hằng ngày của con người.’



Trong khoảng những năm gần đây (từ năm 2000), điện thoại di động ngày càng trở nên quen thuộc với cuộc sống của con người (thế giới nói chung và Việt Nam nói riêng).

Tại Việt Nam, với sự đa dạng các nhà cung cấp, sóng điện thoại được phủ sóng toàn quốc, việc sử dụng điện thoại di động để liên lạc trong kinh doanh, hoạt động hằng ngày đã trở nên phổ biến. Điện thoại di động với ưu điểm gọn nhẹ, dễ sử dụng đã trở thành công cụ thiết yếu của mọi người, nhất là giới trẻ và danh nhân.

Theo số liệu thống kê của Tổng cục thống kê, trong năm 2010, cả nước có 170,1 triệu thuê bao điện thoại, trong đó thuê bao cố định chỉ có 16,4 triệu, còn lại là **154 (90.5%)** triệu thuê bao di động, với tỉ lệ tăng trưởng là **35%***.* Đây là một con số rất ấn tượng và theo dự đoán, con số này sẽ tăng trưởng nhanh trong những năm tới.

Một cuộc khảo sát về tỉ lệ người Việt Nam sử dụng điện thoại di động thực hiện bởi công ty nghiên cứu thị trường Nielsen, kết quả cho thấy Việt Nam có tỷ lệ người sử dụng điện thoại di động cao hơn cả Trung Quốc và Ấn Độ. Trong 5.000 người tại các thành phố và ngoại thành của Việt Nam có 58% dân thành thị và 37% dân khu vực ngoại thành có điện thoại di động riêng. Tại TP.HCM và Hà Nội, tỷ lệ này lên đến 74%. Những con số này còn cao hơn Trung Quốc và Ấn Độ với tỷ lệ dân số sở hữu ĐTDĐ lần lượt là 46% và 30%. Kết quả khảo sát cho thấy đa số người Việt Nam trong độ tuổi từ thanh thiếu niên đến 60 tuổi đều có một thậm chí hai chiếc ĐTDĐ.

Với mức độ tăng trưởng kinh ngạc như vậy, không còn lạ gì khi Việt Nam đứng thứ **27** trong danh sách các quốc gia có số lượng người sử dụng điện thoại di động nhiều nhất trên thế giới.

### Smartphone và thị trường lớn dành cho ứng dụng điện thoại di động



Trước năm 2007, tỉ lệ sử dụng smartphone (điện thoại di động thông minh) tại Việt Nam hầu như rất thấp và smartphone được sử dụng hầu hết bởi danh nhân và được coi là sản phẩm cao cấp.

Nhưng từ khi Apple tung iPhone ra thị trường, thế giới gần như bị ảnh hưởng bởi làn sóng smartphone mới, đa ứng dụng và có thể kết nối Internet. Sự xuất hiện của iPhone như một sự thức tỉnh với các hãng sản xuất điện thoại di động khác. Các hãng sản xuất đua nhau đưa các dòng sản phẩm smartphone khác nhau, làm cho smartphone dần trở thành một sản phẩm công nghệ thông dụng, thay thế dần dòng điện thoại di động truyền thống, vốn chỉ sử dụng để gọi, nhắn tin là chủ yếu.

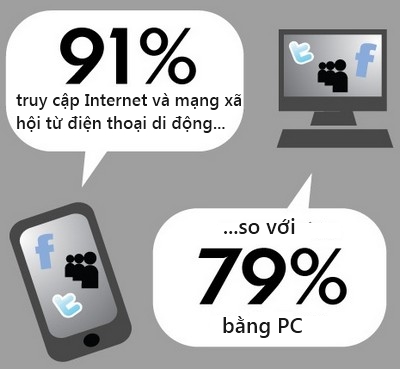
Với dòng smartphone hiện đại kết hợp với sự phát triển vũ bão của Internet và mạng xã hội, điện thoai di động không còn chủ yếu để gọi và nhắn tin nữa, mà còn đảm nhiệm nhiều nhiệm vụ khác như soạn văn bản, lưu trữ hình ảnh, video, chống trộm, máy chơi game,…và nhất là dùng để truy cập Internet.

Dưới đây là một vài con số ấn tượng về tình trạng sử dụng điện thoại di động trên thế giới:

- Hiện nay, toàn thế giới có khoảng 4 tỉ người sử dụng điện thoại di động. Trong đó có 1,08 tỉ người sử dụng smartphone, còn 3,05 tỉ người sử dụng các loại điện thoại cơ bản (nghe, gọi và nhắn tin). Trong đó, có đến khoản 950 triệu người sử dụng các loại điện thoại không có chức năng nhắn tin.

- 86% người sử dụng điện thoại di động truy cập Internet trong khi đang xem TV.

- Hiện nay, có đến 91% lượng truy cập vào các mạng xã hội đến từ các thiết bị di động, nhiều hơn so với 79% lượng truy cập đến từ máy tính cá nhân.



- Theo thống kê, 61% người sử dụng điện thoại di động để chơi game, 55% để theo dõi tình hình thời tiết, 50% sử dụng để tìm kiếm và dò bản đồ, 49% sử dụng để truy cập mạng xã hội, 42% dùng để nghe nhạc, 36% theo dõi tin tức, 33% dùng để xem phim…

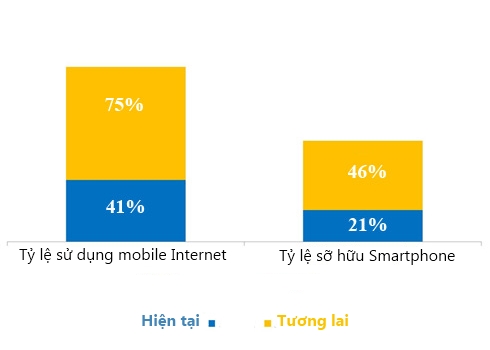
- 30% số người sử dụng smartphone truy cập các mạng xã hội thông qua trình duyệt mặc định của di động. Còn lại cài đặt thêm các trình duyệt khác để sử dụng.

- Mỗi ngày, có đến hơn 200 triệu lượt xem video từ các thiết bị di động.

Tại Việt Nam, số lượng người sử dụng mobile Internet (sử dụng điện thoại di động để truy cập và sử dụng các ứng dụng trực tuyến) với tỉ lệ còn thấp. Kết quả khảo sát gần đây nhất cho thấy chỉ 2 trong số 5 người Việt Nam là sử dụng mobile Internet.

Tuy nhiên con số này sẽ bùng phát nhanh chóng trong vòng 12 tháng tới. Nhận định này được hãng nghiên cứu thị trường Nielsen đưa ra trong báo cáo mang tên **Người tiêu dùng công nghệ số khu vực Đông Nam Á**.

Theo Nielsen, phải mất thời gian khá dài mobile Internet mới có thể tiếp cận được Việt Nam nhưng tới đây, lượng người dùng sẽ gia tăng nhanh chóng. Hiện tại, chỉ 41% người dùng Internet vào mạng qua điện thoại di động, tỷ lệ thấp nhất trong khu vực Đông Nam Á. Nhưng do xu hướng thích ứng nhanh với công nghệ mới của người Việt Nam, tỷ lệ này trong vòng 12 tháng tới sẽ là 75%.



Hình ‑ - Dự đoán tỉ lệ sử dụng smartphone và mobile Internet trong năm tới

Theo đó 96% người dùng Internet đều đã truy cập vào ít nhất một trang mạng xã hội. 79% trong số họ từng bày tỏ thái độ yêu thích (like) hoặc dõi theo một nhãn hiệu, một công ty hoặc người nổi tiếng nào đó trên mạng. Tỷ lệ này cao hơn 10% so với mức bình quân của khu vực Đông Nam Á.

Người dùng Internet Việt Nam cũng dễ bị ảnh hưởng bởi những thông tin tìm thấy trên mạng xã hội. 58% những ý kiến đăng tải được cho là tin cậy và 81% sử dụng mạng xã hội như một nguồn lực để ra các quyết định mua sắm.

Với những con số đầy hứa hẹn như trên, ***có thể thấy một tương lại rộng mở cho thị trường ứng dụng cho smartphone tại Việt Nam***. Với sự phát triển của công nghệ và Internet, một ngày không xa, smartphone sẽ hoàn toàn thay thế cho máy tính cá nhân trong việc liên lạc, làm việc của người dân Việt Nam cũng như thế giới.

### Hệ điều hành Android – Sự bùng nổ của Smartphone.

****

**Android là gì?**

Android là một hệ điều hành ***mã nguồn mở*** dành cho các thiết bị di động như smartphone, tablet,.. Được phát triển bởi Open Handset Alliance (đứng đầu bởi Google). Android được phát triển dựa trên nhân Linux, kết hợp với các thư viện, middleware và API được viết bằng ngôn ngữ C, ứng dụng trên Android được thực thi trên một Application Framework, trong đó có hỗ trợ các ứng dụng Java. Android sử dụng một Java Virtual Machine, Dalvik, để chạy các ứng dụng được viết bằng Java.

**Tại sao lại là Android?**

Android được cung cấp dưới dạng một phần mềm mã nguồn mở, hoàn toàn miễn phí. Điều này có nghĩa các nhà sàn xuất thiết bị di động (smartphone, tablet) có thể sử dụng android và tùy chỉnh theo ý thích.

Nếu như trước đây, mỗi một nhà sản xuất thiết bị di động sử dụng hệ điều hành của riêng mình, kho ứng dụng cũng chỉ phục vụ cho riêng hệ điều hành đó, làm giới hạn khả năng lựa chọn linh hoạt của người sử dụng thì nay người sử dụng có thể lựa chọn sử dụng smartphone với nhiều model, giá cả hợp túi tiền và điều hay nhất là có thể tận dụng được số lượng lớn ứng dụng dành cho Android trên Internet mà không cần quan tâm tới hãng sản xuất. Tất cả là nhờ Android. Đây có thể coi là thế mạnh của Andoird so với iOS của Apple.

**Sự bùng nổ Android trên thị trường**

* Ngày 15/12/2009: số lượng ứng dụng dành cho Android đã đạt mốc 20.000. Chỉ 10 tháng kể từ khi Android lần đầu tiên xuất hiện (2008) và và 3 tháng từ khi đạt mốc 10.000 ứng dụng (9/2009). Ứng dụng dành cho Android đang phát triển với tốc độ rất nhanh. Tại thời điểm 7/2011, tổng số lượng ứng dụng của Android là 250.000, một con số khổng lồ.
* Hệ điều hành Android được hơn 36 nhà sản xuất thiết bị di động sử dụng, trong đó có các ông lớn như HTC, ASUS, Acer, Samsung, LG, Nokia, DELL,..
* Tại thời điểm hiện tại, có hơn 310 mẫu thiết bị đang được lưu hành toàn cầu sử dụng hệ điều hành Android.
* Google đã kích hoạt hơn 100 triệu thiết bị sử dụng hệ điều hành Android trên toàn cầu.

**Hướng phát triển của Android**

Do thế mạnh là một phần mềm mã nguồn mở, hệ điều hành Android có thể được sử dụng trên nhiều loại thiết bị khác chứ không chỉ riêng Smartphone.   
Ngoài smartphone ra, trên thị trường thế giới đã xuất hiện các thiết bị dành cho gia đình sử dụng Android

* *Android TV*: Google đã thông báo chính thức sẽ phát triển TV sử dụng hệ điều hành Android (Google TV). Sản phẩm này có thể tương tác với người dùng, kết nối Internet và sử dụng ứng dụng của Android như smartphone.
* *Android Tablet*: Máy tính bảng sử dụng Android không còn lạ lẫm khi các hãng sản xuất laptop lớn như Dell, Acer, ASUS, LG đều đã tung ra các mẫu sản phẩm để cạnh trang với iPad của Apple.
* *Android Home Phone System*: hệ thống điện thoại cố định sử dụng Android được trang bị trong nhà. Các điện thoại này được kết nối Internet và thực hiện các cuộc gọi VoIP với nhau.

*Android Appliances*: không chỉ xuất hiện ở thiết bị giải trí, di động. Một số hãng sản xuất đã sử dụng Android lên các thiết bị gia dụng bao gồm máy giặt, máy sấy và kể cả lò vi sóng được điểu khiển bằng hệ điều hành Android.



Hình ‑ - Các loại thiết bị sử dụng hệ điều hành Android

**Phát triển ứng dụng trên Android có dễ dàng?**

Với thế mạnh hỗ trợ ứng dụng Java, việc phát triển ứng dụng Android trở nên rất dễ dàng khi mà đã có sẵn một lượng lớn library mã nguồn mở dành cho Java có rất nhiều trên Internet. Developers có thể rút ngắn rất nhiều thời gian để phát triển ứng dụng trên nền Android.

## NHU CẦU

### Phát triển các ứng dụng thuần Việt dành cho Android

Với sự phát triển bủng nổ của hệ điều hành Android và các thiết bị sử dụng hệ điều hành này, có thể nói phát triển ứng dụng Android là một thị trường màu mỡ đang lên không kém gì thị trường ứng dụng dành cho Windows.

Số lượng ứng dụng 250.000 ứng dụng trên Android Market có thể coi là một con số lớn, nhưng đó là những ứng dụng ở nước ngoài. Ở Việt Nam, smartphone sử dụng Android đã xuất hiện và làm mưa làm gió nhưng tỉ lệ người sử dụng so với điện thoại truyền thống vẫn chưa cao. Tuy nhiên, con số này sẽ thay đổi nhanh chóng trong một vài năm tới, khi mà smartphone trở nên nhiều tính năng hơn nữa, và giá thành hợp lý với người dân Việt Nam.

Tại thời điểm mà Android trở nên gần gũi với người dân Việt Nam, nhu cầu sử dụng phần mềm thuần Việt như từ điển, ứng dụng liên lạc, kết nối mạng xã hội, trò chơi, ứng dụng văn phòng,…sẽ gia tăng nhanh chóng. Do vậy, trong thời gian từ bây giờ cho tới thời điểm đó, các công ty ở Việt Nam phải có chiến lược để nghiên cứu và phát triển ứng dụng thuần Việt, hướng tới người Việt dành cho Android để cạnh tranh với các ứng dụng của công ty nước ngoài.

### Nhu cầu ứng dụng từ điển dành cho Android

Nhu cầu sử dụng Android ngày càng tăng, số người sử dụng Android để thay thế cho máy tính cá nhân trong việc làm và học tập ngày càng tăng. Một trong những lợi ích mà máy tính cá nhân làm xuất sắc đó là trở thành từ điển số với ưu điểm nhiều từ, tra cứu nhanh, cập nhật từ dễ dàng. Nay với Android, chiếc điện thoại sẽ trở thành một từ điển bỏ túi cực kỳ tiện dụng với nhiều tính năng hơn từ điển truyền thống.

Qua thời gian tìm hiểu, nhóm thực hiện đề án nhận thấy số lượng ứng dụng từ điển cho Android có rất nhiều nhưng ứng dụng từ điển thuần Việt thì ít và phải trả phí bản quyền (Lacviet mtdEVA for Android). Một thực tế khác là các định dạng từ điển mở được biên soạn kỹ và hỗ trợ tiếng Việt sang các ngôn ngữ khác có rất nhiều trên Internet. Do đó, nhóm thực hiện đề án quyết định xây dựng một ứng dụng từ điển dành cho hệ điều hành Android với các mục tiêu sau:

* Hoàn toàn miễn phí cho người sử dụng.
* Đáp ứng đủ chức năng mà một ứng dụng có phí có.
* Hỗ trợ các định dạng từ điển phổ biến để tận dụng lượng từ điển miễn phí đồ sộ được chia sẽ trên Internet.
* Hoạt động nhanh, dễ sử dụng.
* Có hỗ trợ phát âm.

## CÁC THÁCH THỨC

### Cần thời gian để tìm hiểu về kiến trúc và cách sử dụng API của Android

Android là một nền tảng mới, đòi hỏi thời gian nghiên cứu tài liệu và tiến hành học và thử nghiệm API trước khi bắt tay vào thực hiện phát triển ứng dụng.

Kiến trúc của Android được xây dựng hoàn toàn mới và cung cấp một thư viện widget dồi dào và nhiều thành phần khác. Android định nghĩa một số khái niệm mới về Intent và Activity để lập trình viên có thể tạo và sử dụng các thành phần GUI có sẵn.

### Không có khả năng mua thiết bị Android thật sự

Tại thời điểm hiện tại, giá thành của một smartphone có trang bị hệ điểu hành Android khá cao, cho nên nhóm thực hiện đề án hầu như phát triển và test trên môi trường giả lập được cung cấp sẵn kèm bộ SDK.

Nhược điểm của việc chỉ phát triển và chạy thử nghiệm trên trình giả lập (emulator) là :

Tiêu tốn nhiều thời gian hơn khi thực hiện unit testing: mỗi một lần test là một quy trình đóng gói thành ứng dụng, cài đặt lên trình emulator và chạy thử. Cho nên việc thực hiện unit testing trên Android rất tốn thời gian và khó ghi nhận lại nếu như có lỗi.

Không biết được chính xác ứng dụng thực thi trên nền Android thật sự như thế nào. Vì Android là một hệ điều hành mã nguồn mở, mỗi một nhà cung cấp điện thoại có thể tùy chỉnh theo ý muốn, cho nên việc ứng dụng chạy tốt trên emulator có thể sẽ không chạy trên một số thiết bị thật sự.

### Giới hạn khả năng vật lý của thiết bị

Phát triển ứng dụng trên môi trường thiết bị di động thường sẽ phải đối mặt với các vấn đề sau:

* CPU tốc độ không cao
* Bộ nhớ RAM thấp
* Tốc độ thực thi
* Khả năng đồ họa
* Kích cỡ màn hình khác nhau

Các smartphone có Android đều hỗ trợ multi-tasking và giao diện GUI chú trọng hình ảnh đẹp mắt, cho nên lượng tài nguyên sử dụng cho GUI thường rất nhiều. Do đó lập trình viên phải tìm cách tối ưu hóa các giải thuật xử lí sao cho sử dụng bộ nhớ RAM và tốc độ CPU hợp lí nhất.

### Số lượng khổng lồ các ứng dụng Android trên Internet

Số lượng các ứng dụng Android trên Android Market đã lên đến con số 250.000, nghĩa là nếu như muốn phát triển một ứng dụng Android thì trước hết sẽ đối mặt với một thử thách là ứng dụng đó đã có sẵn và hoạt động ổn định hoặc rất tốt. Điều muốn nói ở đây đó là sự thách thức phải làm sao cho ứng dụng của mình có điểm nổi trội hơn các ứng dụng sẵn có hoặc ít nhất là hoạt đổng tốt ngang bằng.

## XÁC ĐỊNH ĐỀ TÀI



Như đã đề cập ở phần Nhu Cầu, nhóm thực hiện đề án quyết định tìm hiểu và xây dựng ứng dụng từ điển hoạt động trên nền tảng Android, đặt tên là **MEGADICT** với các tiêu chí sau:

* Hoàn toàn miễn phí cho người sử dụng.
* Đáp ứng đủ chức năng mà một ứng dụng có phí có.
* Hỗ trợ các định dạng từ điển phổ biến để tận dụng lượng từ điển miễn phí đồ sộ được chia sẽ trên Internet.
* Hoạt động nhanh, dễ sử dụng.
* Có hỗ trợ phát âm.

### Mục tiêu cơ bản

1. **Tra cứu từ**

Người sử dụng nhập vào từ khóa cần tra và megadict sẽ trả về kết quả là định nghĩa hoặc nội dung mà từ khóa đó được định nghĩa bởi từ điển mà người dùng cài đặt. Người sử dụng có thể chọn lựa tìm kiếm trên một hay nhiều từ điển cùng một lúc, phụ thuộc vào bao nhiêu từ điển mà người dùng cài đặt.

Từ điển là định dạng mà megadict hỗ trợ và được đặt trong thư mục cố định (các phiên bản kế tiếp sẽ hỗ trợ nhiều định dạng và cho phép người sử dụng chỉ định thư mục chứa từ điển).

Nội dung từ điển có thể là từ điển ngôn ngữ, từ điển khoa học, từ điển đồng nghĩa, hệ thống số điện thoại,..Bất cứ cơ sở dử liệu nào hoạt động theo phương thức từ khóa – nội dung.

1. **Phát âm từ**

megadict tận dụng sức mạnh của chức năng *Text-to-speech* được cài đặt theo thiết bị sử dụng Android để phát âm các từ khóa. Ngôn ngữ hỗ trợ phụ thuộc vào thiết bị và phiên bản hệ điều hành Android sử dụng. Người sử dụng phải cài đặt và bật các *Text-to-speech engine* thì mới có thể sử dụng được. Các ngôn ngữ hỗ trợ bởi *Text-to-speech* là : English (tiếng Anh), French (tiếng Pháp), German (tiếng Đức), Italian (tiếng Ý) và Spanish (tiếng Tây Ban Nha).

Tận dụng sức mạnh của smartphone, tốc độ mạnh mẽ của đường truyền Internet (mà chủ yếu là công nghệ 3G) cùng với dịch vụ miễn phí của *Google Translate*, megadict cho phép mở rộng khả năng phát âm từ với nhiều loại ngôn ngữ hơn Text-to-speech của Android. Số lượng ngôn ngữ được hỗ trợ lên đến hơn 50 ngôn ngữ, có cả Tiếng Việt.

1. **Hỗ trợ một loại định dạng tài liệu thông dụng**

Một trong những tiêu chí chính của megadict là tận dụng lượng từ điển miễn phí được cung cấp trên Internet, nên việc hỗ trợ loại định dạng tài liệu thông dụng là rất cần thiết.

Hiện tại các loại định dạng từ điển có rất nhiều, nhưng lượng từ điển miễn phí chủ yếu có các định dạng:

* + *DICT*: đây là định dạng được phát triển bởi DICT Development Group nhằm tạo ra một loại định dạng từ điển có thể vượt qua Webster protocol để truy cập nhiều từ điển đồng thời.
  + *StarDict*: định dạng chính của từ điển mã nguồn mở StarDict, được phát triển từ DICT, nhưng được thiết kế lại để phục vụ cho một vài mục đích nhất định.
  + *XDXF*: định dạng sử dụng XML để định nghĩa từ khóa và nội dụng của từ khóa. XDXF được xây dựng nhằm mục đích hội tụ tất cả các loại định dạng từ điển khác với nhau.

Do thời gian của đề tài có hạn, nhóm thực hiện dự án quyết định hỗ trợ định dạng DICT trước tiên, vốn có một thời gian tồn tại lâu và tổng số từ điển có thể tìm thấy lớn.

1. **Ứng dụng thực thi nhanh**

Nhóm thực hiện đề án quyết định tốc độ thực thi là một mục tiêu quan trọng mà megadict cần đạt được. Do hỗ trợ tìm kiếm trên nhiều từ điển cùng lúc và nhiều loại từ điển khác nhau, cho nên megadict cần phải bảo đảm tốc độ thực thi là nhanh nhất có thể. Nếu một từ điển hoạt động chậm thì sẽ không có người dùng nào kiên nhẫn sử dụng nó cả.

1. **Dễ sử dụng**

Megadict được thiết kế theo tiêu chí: chỉ-nhìn-là-biết-dùng-ngay, người sử dụng có thể nhìn vào là hiểu ngay cách sử dụng. Các chức năng chính được trình bày ngay ở giao diện chính, để người dùng không cần phải suy nghĩ cũng như cũng không cần hướng dẫn sử dụng quá rườm rà.

### Mục tiêu nâng cao

1. **Hỗ trợ thêm nhiều định dạng từ điển khác**

Hỗ trợ nhiều loại từ điển khác nhau đồng nghĩa với việc người sử dụng có nhiều lựa chọn hơn trong việc tìm kiếm và sử dụng nguồn từ điển miễn phí trên Internet.

Hỗ trợ nhiều loại từ điển cũng có nghĩa megadict không phụ thuộc vào một loại từ điển nhất định. Các loại từ điển sẽ hỗ trợ tiếp theo là các loại từ điển đã liệt kê ở phần Mục tiêu cơ bản.

Ngoài ra, nếu như tiếp tục phát triển, nhóm thực hiện có dự định thiết kế và cài đặt một loại định dạng từ điển mới có thể tối ưu tốc độ tìm kiếm và đọc dữ liệu để tăng hiệu suất hoạt động.

1. **Đảm bảo tốc độ thực thi nhanh, ổn định**

Như đã đề cập ở tiêu chí, tốc độ thực thi nhanh luôn là mục tiêu quan trọng của megadict. Nhóm thực hiện muốn tốc độ của megadict luôn được bảo đảm cho dù số lượng từ điển mà người sử dụng cài đặt là bao nhiêu đi nữa.

1. **Tương tác với các ứng dụng khác trong Android**

Nhóm thực hiện đề án muốn megadict trở thành một từ điển chạy ngầm, có thể được gọi bất cứ lúc nào, bởi bất cứ ứng dụng nào trên Android. Nghĩa là khi người sử dụng lướt web bằng 1 browser, hay xem tài liệu bằng một ứng dụng xem văn bản, hay bất cứ ứng dụng nào có liên quan đến văn bản, khi gặp một từ hoặc một vấn đề gì đó cần tra cứu, có thể bật megadict để tra cứu với tổ hợp phím hoặc phương thức nào đó người sử dụng tự định nghĩa.

### Hướng phát triển

1. **Phát triển trở thành từ điển tiêu biểu cho người Việt Nam sử dụng Android**

Nếu nhắc đến từ điển sử dụng trên Windows của người Việt, người ta nghĩ ngay tới LacViet mtdEVA. Nhóm thực hiện cũng mong muốn khi nghĩ tới một ứng dụng từ điển dành cho Android, thì megadict là lựa chọn đầu tiên được nghĩ tới. Mọi người tin tưởng và sử dụng megadict, giới thiệu những người khác sử dụng megadict. Các thiết bị android ở Việt Nam được cài đặt sẵn megadict.

1. **Mở rộng ứng dụng trên các platform khác**

Android không phải là hệ điều hành duy nhất dành cho smartphone. Thị trường smartphone cực kỳ rộng lớn và có nhiều hệ điều hành khách cũng không kém thị phần như iOS (Apple), Bada (Samsung), Symbian, Windows Phone,…Mục tiêu kế tiếp là megadict được implement để mở rộng sang các nền tảng này để người sử dụng tin dùng megadict không cần phải chọn lựa một ứng dụng từ điển khác nếu như họ muốn đổi smartphone đang sử dụng.

# CÁC KIẾN THỨC LIÊN QUAN

## Hệ điều hành di động (mobile operating systems)

### Giới thiệu

Hệ điều hành di động (*Mobile operating system*) được biết như *mobile OS*, *mobile software* platform hay *handheld operating system*, là hệ điều hành điều khiển thiết bị di động giống như các hệ điều hành dành cho máy tính để bàn hay laptop như Windows, Mac OS X hay Linux. Tuy nhiên, chúng đơn giản hay, xử lý nhiều hơn về vấn đề wiereless, kết nối nội bộ, các định dạng multimedia và những giải pháp đầu vào khác nhau.

Những ví dụ tiêu biểu cho các thiết bị chạy trên hệ điều hành di động là smartphones, personal digital assistants (PDAs), tablet computer và information applicance, hoặc đôi khi ám chỉ các thiết bị thông minh có thể dùng với các hệ thống nhúng, hay các thiết bị di động khác và thiết bị không dây.

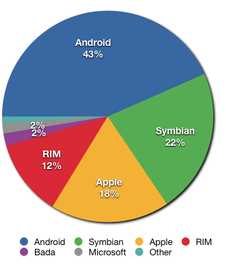


Hình ‑- Các hệ điều hành di động

### Lịch sử

Tầm quan trọng ngày càng tăng của thiết bị di động đã gây ra sự tranh đua mãnh liệt giữa các nhà khổng lồ về phần mềm như Google, Microsoft và Apple, cũng như những công ty hàng đầu về thiết bị di động như Nokia, Research In Motion (RIM) và Palm, trong nỗ lực nắm bắt được thị trường lớn nhất này.

Với sự phát hành của iPhone vào năm 2007, Apple đã tấn công vào ngành công nghiệp di động và mở ra một kỷ nguyên của hệ điều hành dành cho smartphone, tập trung vào kinh nghiệm của người dùng và trông cậy vào tương tác hướng chạm. 10/2007, Google thành lập Open Handset Alliance với 79 công ty viễn thông, phần cứng, phầm mềm khác nhau để xâm nhập vào thị trường smartphone thông qua hệ điều hành Android. Mặc dù nhận được những phản hồi tích cực từ cộng đồng và giới truyền thông, phiên bản đầu của Android đã tạo ra vết nứt giữa Apple và Google, hậu quả là dẫn tới sự từ chức của phó tổng giám đốc của Google, Eric Schmidt, ra khỏi ban giám đốc của Apple.



Hình ‑ -Thị phần smartphone được bán ra theo hệ điều hành vào quý 2, 2011 được phân tích bởi Gartner

Từ khi có sự xuất hiện của iOS của Apple và Android của Google, thị trường smartphone đã bùng nổ thật sự và tháng 5/2010 đã giải thích cho hơn 17,3% thiết bị di động đã được bán ra. Điều này dẫn tới sự nhận thức càng lớn của người tiêu dùng các hệ điều hành di động, khiến cho các nhà sản xuất quảng cáo liên tục về những hệ điều hành di động khác nhau của họ. 2/2011, Google giữa 33,3% thị phần smartphone trên toàn thế giới, cho thấy sự lớn mạnh đáng kinh ngạc của Android OS so với năm trước đó là 4,7%. Nokia, Apple, RIM và Microsoft tương ứng giữ 31%, 16,2%, 14,6% và 3,1% thị phần.

### Dự đoán về thị phần

Hệ điều hành di động đang trong giai đoạn non trẻ và thật khó để dự đoán về sự phát triển thị phần trong tương lại. Tuy nghiên, một xu hướng rõ ràng là sự phát triển tràn lan của các hệ điều hành được phát triển cho thiết bị thông minh hơn là cho những điện thoại thông thường. 2/2011, Nokia đã thông báo một cộng tác với Micrsoft đã kết thúc sự phát triển của Symbian OS, hệ điều hành phổ biến nhất dành cho điện thoại thông thường vào cuối 201.

Đáng chú ý rằng Intel đang bắt đầu tập trung vào các thiết bị di động hơn là điện thoại di động. Có thể kể đến là Mobile Internet Devices (MID) và Ultra-Mobile PC (UMPC). Trong lúc đó, Palm đã từ bỏ kế hoạch phát triển Foleo, một thiết bị đi chung với smartphone.

### Các hệ điều hành phổ biến

#### Android của Google Inc. (mã nguồn mở)

Android được phát triển bởi một công ty được mua bởi Google Inc., và Google tiếp tục cập nhật phần mềm. Android là hệ điều hành mã nguồn mở dẫn xuất từ Linux kernel cùng với những nhà phát triển phần mềm như Intel, HTC, ARM, Samsung. Motorola và eBay, đã định hình Open Handset Alliance.



Hình ‑ - Samsung Galaxy Mini chạy HĐH Android

#### BlackBerry của RIM (mã nguồn đóng)

BlackBerry tập trung vào thao tác dễ dàng và được thiết kế cho việc kinh doanh. Gần đây rất nhiều những ứng dụng bên thứ 3 đã được phát triển và BlackBerry đã được nâng cấp để hỗ trợ đầy đủ cho multimedia. Chiến lược tương lai của RIM là tập trung vào hệ điều hành QNX và trông đợi QNX dành cho smartphone sẽ ra mắt vào đầu năm 2012.



Hình ‑ - BlackBerry Torch 9810 chạy HĐH BlackBerry

#### iOS của Apple Inc. (mã nguồn đóng)

iPhone, iPod hay iPad, tất cả đều sử dụng chung một hệ điều hành được gọi là iOS, cái được dẫn xuất từ Mac OS X. Những ứng dụng của tổ chức thứ 3 không được hỗ trợ chính thức cho tới phiên bản iOS 2.0 vào 7/2008. Trước đó, jailbreaking sẽ cho phép cài ứng dụng và giải pháp này hiện nay vẫn còn. Hiện tại tất cả thiết bị dùng iOS được phát triển bởi Apple và được sản xuất bởi Foxconn hoặc những đối tác của Apple.



Hình ‑ - iPad chạy HĐH iOS

#### Symbian của Symbian Foundation (open public license)

Symbian có thị phần lớn nhất trên hầu hết thị trường, nhưng bị bỏ lại sau những công ty nhỏ khác ở thị trường Bắc Mỹ. Điều này tương đồng với sự thành công của Nokia trong hầu hết thị trường trừ Nhật Bản. Ở Nhật, Symbian rất mạnh bởi sự quan hệ với NTT DoCoMo, với duy nhất một trong 44 máy thu phát cầm tay được tung ra ở nhật từ Nokia. Nó được dùng bởi những nhà sản xuất máy thu phát cầm tay gồm BenQ, Fujitsu, LG, Mitsubishi, Motorola, Nokia, Samsung, Sharp và Sony Ericsson. Những thiết bị dựa trên Symbian hiện nay được tạo bởi Fujitsu, Nokia, Samsung, Sharp và Sony Ericsson. Gần đây, mặc dù sự xuất khẩu thiết bị Symbian đã tăng nhưng thị phần trên toàn thế giới đã sụt giảm từ hơn 50% thành chỉ hơn 40% từ 2009 sang 2010. Nokia đã chuyển giao Symbian cho Accenture để tiếp tục hỗ trợ cho tới 2016.



Hình ‑ - Nokia chạy HĐH Symbian

#### Windows Phone của Microsoft (mã nguồn đóng)

15/2/2010, Microsot khai trương hệ điều hành thế hệ tiếp theo là Windows Phone 7. Hệ điều hành mới này sở hữa một giao diện cuốn hút lấy cảm hứng từ Metro Design Language của Microsoft. Nó được tích hợp đầy đủ các dịch vụ của Microsoft như Windows Live, Zune, Xbox Live và Bing, nhưng cũng tích hợp nhiều những dịch vụ không phải của Microsoft như Facebook hay Google accounts. Phiên bản mới nhận được phản hồi tích cực từ báo chí công nghệ.



Hình ‑ - HĐH Windows phone

#### webOS của HP

webOS là hệ điều hành di động độc quyền chạy trên Linux kernel, được phát triển bởi Palm, sau đó được mua bởi HP. HP đã tung ra hai điện thoại là Veer, Pre 3 và tablet TouchPad chạy trên webOS vào năm 2011 trước khi từ bỏ.



Hình ‑ - Tablet HP chạy HĐH webOS

Với số liệu từ , có thể thấy rằng Android đang lên như một cơn sóng, trở thành một hệ điều hành mã nguồn mở dành cho những ai đam mê tìm hiểu và khám phá.

## Giới thiệu về hệ điều hành Android

Android là hệ điều hành dành cho các thiết bị di động như smartphones hay tablet, được phát triển bởi Open Handset Alliance (OHA). OHA là một liên minh doanh nghiệp gồm 83 thành viên tập trung phát triển các tiêu chuẩn cho thiết bị di động. Các thành viên bao gồm: Google, HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Intruments, Samsung, LG, T-Mobile, Nvidia và Wind River Systems.

Android được xây dựng dựa trên Linux kernel với middleware, libraries, các API được viết bằng C và phần mềm chạy trên một application framework bao gồm những thư viện tương thích với Java dựa trên Apache Harmony. Android sử dụng Dalvik virtual machine với phương pháp biên dịch just-in-time để chạy code Java đã được biên dịch. Android có một cộng đồng những developer viết ứng dụng, chủ yếu viết bằng Java. Hiện tại có hơn 250.000 ứng dụng Android. Android Market là một kho ứng dựng online của Google, người dùng có thể download ứng dụng tại đây.

### Lịch sử sáng lập và hoạt động

Android, In.c được sáng lập tại Palo Alto, California, Mỹ vào 10/2003 bởi Andy Rubin, Rich Miner, Nick Sears và Chris White. Rubin đã phát biểu rằng “Những thiết bị di động thông minh là những cái nhận biết được vị trí và những thiết lập của chúng”.

8/2005 Google đã mua Android Inc. và làm cho nó hoàn toàn được sở hữu bởi công ty con của Google Inc. Những nhân lực chính của Android Inc. gồm Andy Rubin, Rich Miner và Chris White vẫn ở lại công ty để phát triển Android. Không nhiều người biết về Android Inc ở thời điểm được thu mua, nhưng một số trong đó đã thừa nhận kết hoạch tấn công vào thị trường điện thoại cầm tay của Google.

### Các phiên bản Android

Android đã cập nhật liên tục từ phiên bản đầu tiên. Những cập nhật cho hệ đìêu hành và thêm những đặc tính mới. Mỗi phiên bản của hệ điều hành Android được phát triển với một tên gọi dựa trên những món tráng miệng. Code name của chúng được sắp theo thứ tự alphabet: Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb và sắp tới là Ice cream sandwhich. Dưới đây là danh sách các phiên bản gần đây nhất:

+ 2.0 ( Eclair) nâng cấp lên web browser và giao diện mới, hỗ trợ HTML5 và W3C geolocation API. Bản này được cài thêm ứng dụng camera với những đặc tính zoom, flash, color effects và nhiều thứ khác.

+ 2.1 (Eclair) hỗ trợ điều khiển giọng nói, new laucher, 5 homescreen thay vì 3 như trước, background động, một nút để mở menu. Bản này được cài thêm ứng dụng thời tiết, nâng cấp email và phonebook.

+ 2.2 (Froyo) đưa ra sự cải thiệt tốc độ với giải pháp tối ưu JIT và Chrome V8 Javascript engine, thêm vào Wifi hotspot tethering và hỗ trợ Flash.

+ 2.3 (Gingerbread) điều chỉnh lại giao diện, nâng cấp bàn phím mềm, đặc tính copy/paste và hỗ trợ Near Field Communication.

+ 3.0 (Honeycomb) là phiên bản dùng cho tablet, hỗ trợ cho các thiết bị có màn hình lớn hơn và đưa ra nhiều đặc tính giao diện mới, hỗ trợ xử lý đa nhân và tăng tốc phần cứng đồ họa. Thiết bị đầu tiên dùng Honeycomb là Motorola Xoom, được bày bán vào 2/2011.

+ 3.1 (Honeycomb) được thông báo tại Google I/O vào 10/2011, cho phép các thiết bị Honeycomb truyền dữ liệu trực tiếp từ các USB.

+ 3.2 (Honeycomb) là phiên bản cài thêm, có những khả năng mới cho người dùng và developer. Nổi bật gồm có tối ưu làm tăng kích thước màn hình, khả năng “zoom-to-fill”, load media file trực tiếp từ SD card và màn hình mở rộng hỗ trợ developer điều chỉnh giao diện chính xác hơn.

+ 4.0 (Ice Cream Sandwhich) sẽ là sự kết hợp của Gingerbread và Honeycomb.

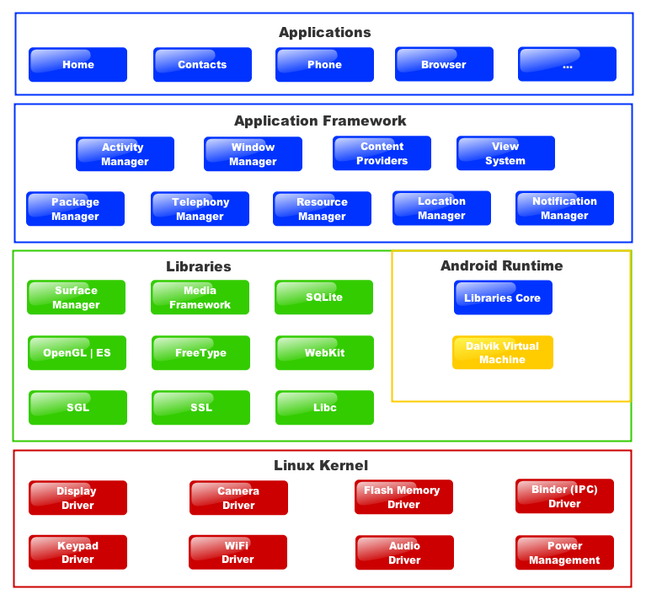
### Kiến trúc Android

Kernel của Android được dẫn xuất từ Linux kernel. Google đã đóng góp code cho Linux kernel như một phần của sự cố gắng, nhưng những tính năng đáng chú ý có thể kể đến là quản lý năng lượng được gọi là wakelocks đã bị từ chối bởi những kernel developer chính thống. Vì lẽ đó Android kernel giờ đây là một phiên bản riêng biệt hoặc là một “người anh em” của Linux kernel.

4/2010, Google thông báo thằng họ sẽ thuê hai công nhân làm việc cho cộng đồng Linux kernel. Greg Kroah-Hartman, hiện tại là người bảo trì Linux kernel ở nhánh đảm bảo sự ổn định, vào 20/2010 đã phát biểu rằng ông ta lo ngại Google sẽ không còn cố gắng thay đổi code trong Linux. Một số Google Android developer gợi ý “Nhóm Android ko còn muốn làm việc đó” bởi vì họ là một nhóm nhỏ và còn nhiều công việc cấp bách hơn phải làm trên Android.

Android không có một X Windows System, cũng ko hỗ trợ GN libraries và điều này làm cho khó khăn trong việc đem các ứng dụng hay các thư viện của GNU/Linux vào Android.

- Các tính năng: Handset layouts, storage, connectivity, messaging, multiple language, web browser, Java support, media support, streaming media support, additional hardware support, multi-touch, bluetooth, video calling, multitasking, voice based feature, Wifi hotspot tethering, screen capture.



Hình ‑ – Kiến trúc của HĐH Android

### Android Market

Android Market là một kho phần mềm online được phát triển bởi Google dành cho các thiết bị Android. Một ứng dụng được gọi là “Market” được cài sẵn trên hầu hết các thiết bị Android và cho phép người dùng truy cập và download ứng dụng được tải lên bởi những developer khác. 12/2010, có khoảng 200.000 games, ứng dụ và widget trên Android Market. 4/2011, Google thông báo có hơn 3 tỉ ứng dụng Android được cài và cuối 6/211 có khoảng 6 tỉ lượt cài ứng dụng từ Android Market. Bản thân hệ điều hành này được cài trên 130 triệu thiết bị.

Những thiết bị tuân theo yêu cầu tương thích của Google được phép cài trước phần mềm mã nguồn đóng Android Market và truy xuất market. Bộ lọc của Market liệt kê những ứng dụng tương thích với thiết bị của người dùng và những devloper đã hạn chế ứng dụng đó tùy theo hãng vận tải hoặc quốc gia vì những lý do thương mại.

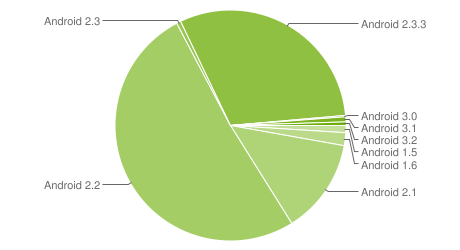
Google đã tham gia vào Android Market bằng cách đưa ra những ứng dụng của riêng họ, bao gồm Google Voice cho Google Voice service, Sky Map để xem các ngôi sao, Finance cho finance service, Maps Editor cho MyMaps service, Places Directory cho Local Search, Google Goggles tìm kiếm hình ảnh, Gesture Search cho những chữ viết tay và số để tìm kiếm nội dung trên điện thoại, Google Translate, Google Shopper, Listen cho podcast và My Tracks một ứng dụng chạy bộ. 8/2010, Google giới thiệu Voice Actions cho Android, cái cho phép người dùng tìm kiếm, viết tin nhắn và gọi điện bằng giọng nói.

Người dùng cũng có thể cài ứng dụng từ kho lưu trữ của tổ chứng thứ 3 như Amazon Appstore hay cài trực tiếp từ APK file.

### Thị phần

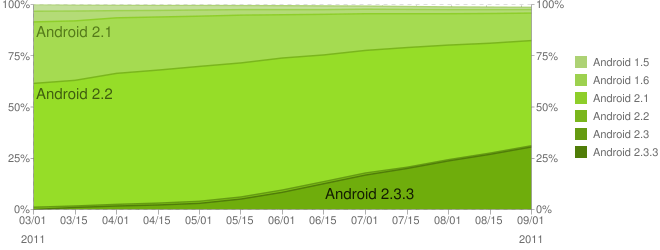
Công ty nghiên cứu Canalys đánh giá trong Q2 2009 rằng Android có 2.8% thị phần trên toàn thế giới ở lĩnh vực smartphone. Theo Q4 2010, tỉ lệ này tăng 33%, trở thành hệ điều hành bán chạy nhất dành cho smartphone. 6/2011, Google phát biểu rằng 550.000 thiết bị Android được kích hoạt mỗi ngày, 440.000 mỗi ngày trong hai tháng tiếp theo vào 5/2011 và hơn 100 triệu thiết bị đã được kích hoạt.

### Thị phần theo phiên bản Android



Hình ‑ - Thị phần theo phiên bản của Android

*(Biểu đồ trên dựa trên số thiết bị Android đã truy xuất Android Market trong chu kỳ 14 ngày kết thúc vào ngày 2/9/2011.)*



Hình ‑ - Thị phần theo phiên bản và theo thời gian của Android

*(Mỗi dataset trong trục thời gian dựa trên số thiết bị đã truy xuất Android Market trong chu kỳ 14 ngày kết thúc vào ngày được báo hiệu trên trục x.)*

## Môi trường lập trình Android

Môi trường lập trình Android phổ biến là dùng Eclipse kết hợp với ADT (Android Development Tool). Cài đặt môi trường lập trình là bước đầu tiên để bước vào thế giới lập trình Android.

### Yêu cầu phần cứng

- RAM tối thiểu là 1GB (1GB đôi lúc sẽ load không nổi emulator), khuyên dùng 2GB RAM trở lên, tôi dùng 3GB mà vẫn lag. Có một chủ đề bàn về tốc độ emulator ở bên dưới.

- Hệ điều hành tôi dùng là Windows, nếu dùng Linux thì tham khảo trên mạng.

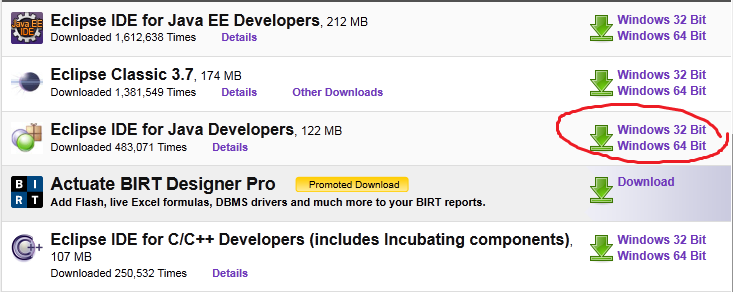
- Card màn hình vừa đủ, cỡ onboard là vừa.

- Ổ cứng cần khoảng 1GB để lưu dữ liệu.

### Cài đặt Eclipse và khởi động emulator

1) Cài eclipse. Vào link sau download “Eclipse IDE for Java developers”.

<http://www.eclipse.org/downloads/>

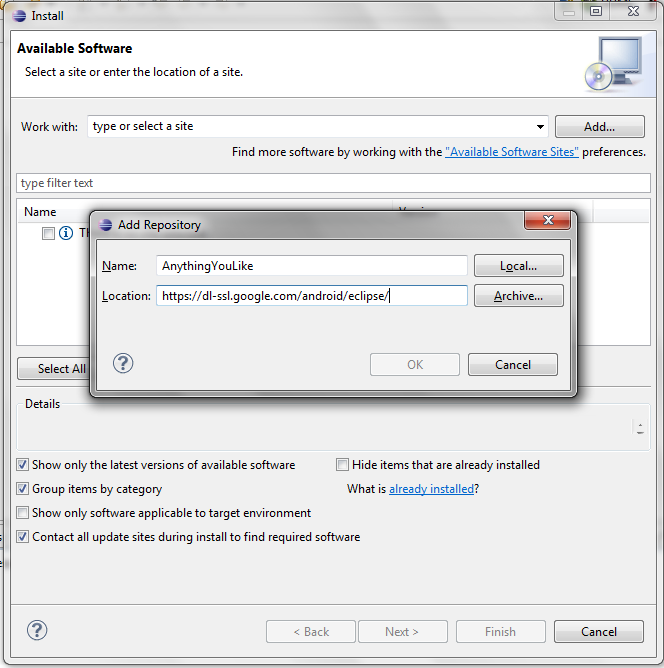


Hình ‑ – Download Eclipse

2) Giải nén file vừa download. Di chuyển thư mục đó vào nơi mong muốn.

3) Đến đây có hai cách cài ADT: online hoặc offline. Bạn hãy chọn một trong hai cách.

- Online: Mở Eclipse, chọn Help / Install new software. Một hộp thoại hiện ra. Nhấn “Add”, hộp thoại “Add repository” hiện ra. Copy link này vào phần “Location”: <https://dl-ssl.google.com/android/eclipse/>. Phần Name gõ tùy thích. Nhấn Ok. Ngồi chờ một lúc. Khi nào có bảng thông báo hiện lên thì nhấn Ok.



Hình ‑ – Cài ADT

- Offline: Vào link này download platform mong muốn.

<http://qdevarena.blogspot.com/2010/05/download-android-sdk-standalone-for.html>

Sau khi download thì giải nén rồi đặt vào đường dẫn “ANDROID\_HOME\ platforms”, với ANDROID\_HOME là thư mục gốc, nơi cài đặt Android SDK.

4) Trở lại Eclipse, vào Windows / Android SDK and AVD Manager. Chọn “New...”, hộp thoại “Create new Android Virtual Device” hiện ra. Ở đây có một số thông số.

- Name: Tên của AVD bạn muốn đặt. Ở đây ta đặt “Android-2.2”.

- Target: Platform hoặc API level bạn mong muốn. VD muốn lập trình trên Android 2.2, chọn “Android 2.2 API level 8”. Ở đây ta chọn Android 2.2.

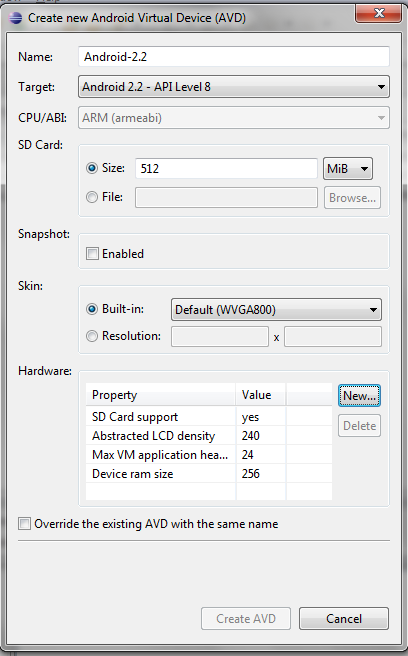
- SD Card: Chọn kích thước mong muốn cho thẻ nhớ. Ta chọn 512MB.

- Snapshot: Snapshot là một dạng lưu trạng thái emulator, khi khởi động lần sau emulator sẽ hiện ngay lập tức. Ở đây tôi không chọn.

- Skin: là bề mặt hay vẻ ngoài của emulator. Để mặc định.

- Hardware: Các thiết lập phần cứng ảo. Đối với các ứng dụng cần dùng SD card thì chọn “New” vào add “SD card support”.

5) Nhấn “Create AVD” để hoàn tất.



Hình ‑ – Thiết lập AVD

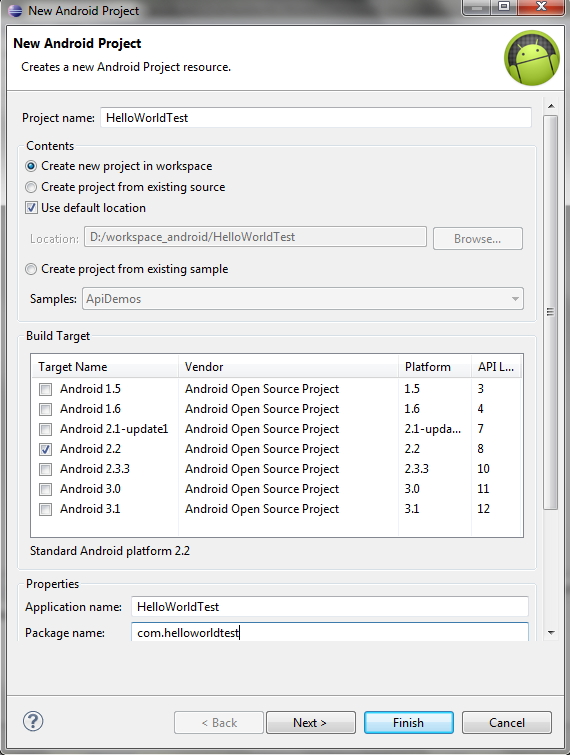
6) Công việc cài đặt đã hoàn tất. Để khởi động emulator, ta chọn “Start…”, có thể điều chỉnh kích thước của emulator ở đây, khi đã xong thì nhấn “Launch”. Đây là màn hình khi emulator đã khởi động xong. Mở khóa bằng cách giữ chuột trên hình ổ khóa, kéo sang phải. (phải mở khóa để chạy ứng dụng ở phần hướng dẫn bên dưới)



Hình ‑ – Emulator đã khởi động xong

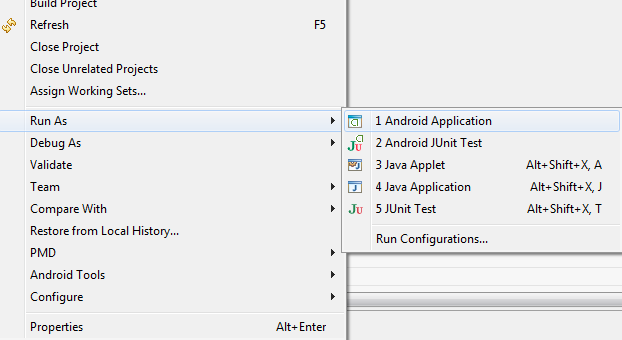
### Tạo Android project và chạy thử

1) Vào Eclipse, chọn menu New / Android Project. Điều chỉnh thông số như trong hình rồi nhấn Finish.

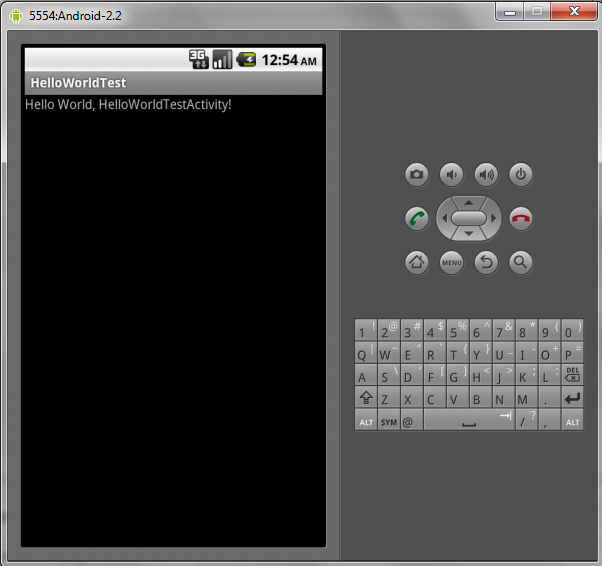


Hình ‑ – Tạo Android project

2) Chuột phải vào project trên Package Explorer hoặc trên Project Explorer, chọn Run As / Android Application. Trở lại màn hình emulator (đã khởi động hoàn tất trước đó), chương trình sẽ hiện hiện lên.



Hình ‑ – Run project



Hình ‑ – Hoàn tất chạy thử Android project

## Android emulator

### Khái quát

Bộ Android SDK của Google đi kèm một bộ mô phỏng thiết bị di động ảo (android emulator) chạy trên PC. Android emulator bắt chước tất cả đặc tính phần cứng và phần mềm của một thiết bị thật, trừ việc không thể thực hiện cuộc gọi thật. Nó cung cấp nhiều cách duyệt và các phím điều khiển, bạn có thể “chạm” bằng cách dùng con chuột hoặc bàn phím. Nó cũng cung cấp màn hình hiển thị ứng dụng, cùng với những ứng dụng khác chạy Android.

Để test ứng dụng dễ dàng, emulator có một chỗ để thiết lập cấu hình gọi là Android Virtual Device (AVD). Những ADVs giúp bạn thay đổi cấu hình phần cứng để có thể test trên nhiều thiết bị khác nhau.

Emulator cũng đi kèm nhiều tiện ích để debug như console để log output của ứng dụng ra ngoài, mô phỏng việc gửi SMS hoặc gọi điện thoại.

Android emulator là ứng dụng dựa trên QEMU, cung cấp một thiết bị di động ảo ARM nơi mà bạn có thể chạy những ứng dụng Android. Bạn có thể chọ phiên bản của hệ thống Android bằng cách thiết lập AVD, cũng có thể customize skin và key mapping. Khi chạy emulator ở thời điểm thực thi, ta có thể dùng những câu lệnh khác nhau để điều khiển hành vi của emulator.

### Android Virtual Devices và Emulator

Để sử dụng emulator, bạn phải tạo một hoặc nhiều thiết lập AVD. Trong mỗi thiết lập, chỉ định một Android platform để chạy emulator và cài đặt những tùy chọn phần cứng và emulator skin mà bạn muốn dùng. Khi chạy emulator, bạn sẽ chọn AVD muốn load.

Mỗi AVD có bộ nhớ riêng của nó. Ví dụ như user data, SD card. Khi chạy emulator với thiết lập AVD, nó tự động load user data và SD card từ thư mục của AVD đó. Mặc định, emulator lưu user data, SD card và cache trong mỗi thư mục AVD.

### Android emulator và những hạn chế

Như đã đề cập ở trên. Android emulator chạy ARM opcode. Nó sử dụng QEMU (một virtualization system na ná như VirtualBox hay VMWare) để mô phỏng thiết bị thật. Cách làm này giúp tối ưu sự đúng đắn và chính xác giữa emulator và thiết bị thật. Cách chạy của emulator là chuyển đổi từ ARM opcode thành Intel opcode. Cách duy nhất để tăng tốc cho emulator là tăng tốc CPU chạy emulator. Tôi vẫn chưa nghiên cứu sâu về emulator nhưng tôi không có cảm giác rằng emulator chạy multiple cores. Vì lý do đó, một máy tính nhanh hơn bị chi phối bởi tốc độ của single core hơn là số core mà nó có.

Để test những ứng dụng đồ họa cao như games và những hiệu ứng hình ảnh, cách tốt nhất là mua thiết bị thật. Còn emulator sẽ giúp ích cho việc test trên nhiều thiết lập khác nhau bằng cách dùng các AVDs.

Kết luận, tuy có nhiều cách điều chỉnh để tăng tốc emulator (ví dụ như chỉnh screen size nhỏ lại, dùng các tham số trong command line) nhưng tốc độ sẽ tăng không đáng kể, đôi khi không có tác dụng. Cách duy nhất và hiệu quả nhất là tăng tốc CPU.

## Lập trình Android

Android là một hệ thống gồm hệ điều hành, middleware và những ứng dụng quan trọng được cài sẵn. Android SDK cung cấp các công cụ và API cần thiết để phát triển ứng dụng trên hệ điều hành Android bằng ngôn ngữ Java.

### Đặc tính

- **Application framework** cho phép tái sử dụng và thay thế uyển chuyển các bộ phận.

- **Dalvik virtual machine** được tối ưu cho các thiết bị mobile.

- **Trình duyệt được tích hợp** dựa trên web engine mã nguồn mở là WebKit.

- **Đồ họa tối ưu** được cung cấp bởi thư viện đồ họa 2D và đồ họa 3D dựa trên OpenGL ES 1.0.

- **SQLite** cho việc lưu trữ dữ liệu trên điện thoại.

- **Hỗ trợ media** cho những định dạng ảnh thông dụng (MPEG4, H.264, MP3, Aac, AMR, JPG, PNG, GIF.

- **Hệ thống gọi điện GSM** (phụ thuộc phần cứng)

- **Bluetooth, EDGE, 3G và WiFi** (phụ thuộc phần cứng)

- **Camera, GPS, la bàn, dụng cụ đo gia tốc** (phụ thuộc phần cứng)

- **Môi trường phát triển màu mỡ** gồm một bộ mô phỏng, công cụ debug, phân tích tốc độ và bộ nhớ, và một plugin dành cho Eclipse.

### Kiến trúc

Sơ đồ sau thể hiện cụ thể những thành phần chính của hệ điều hành Android. Mỗi section được mô tả chi tiết bên dưới.



Hình ‑ - Kiến trúc Android

#### Applications

Android sẽ được đóng gói kèm với một tập hợp ứng dụng cốt lõi gồm email client, ứng dụng gửi SMS, lịch, bản đồ, trình duyện, ứng dụng quản lý contact và những thứ khác. Tât cả ứng dụng được viết bằng ngôn ngữ lập trình Java.

#### Application framework

Bằng cách cung cấp một môi trường phát triển mở, Android cho phép developer xây dựng những ứng dụng đầy sáng tạo và phong phú. Các developer được tự do tận dụng phần cứng của thiết bị, truy xuất vào thông tin vị trí, dịch vụ chạy ngầm, cài đặt báo động, thêm thông báo vào thanh trạng thái, nhiều và nhiều thứ nữa.

Developer có toàn quyền truy cập vào framework API được sử dụng bởi những ứng dụng cốt lõi. Kiến trúc ứng dụng được thiết kế để đơn giản hóa việc tái sử dụng các bộ phận, bất kỳ ứng dụng nào cũng có thể công bố khả năng của nó và bất kỳ ứng dụng khác sau đó có thể sử dụng khả năng đó (chủ thể đối với ràng buộc an toàn là bắt buộc bởi framework). Cơ chế này cho phép những bộ phận được thay thế bởi người dùng.

Bên dưới tất cả ứng dụng là một tập hợp dịch vụ và hệ thống, bao gồm:

- **Một bộ Views** có thể được dùng để xây dựng ứng dụng, gồm list, grid, text box, button và ngay cả những web browser có thể nhúng.

- **Content provider** cho phép ứng dụng truy xuất dữ liệu từ những ứng dụng khác như Contacts, hoặc chia sẻ dữ liệu của riêng chúng.

- **Resource Manager** cung cấp sự truy xuất đến những tài nguyên không phải là code, ví dụ như chuỗi, hình ảnh hay các file layout.

- **Notification Manager** cho phép tất cả ứng dụng thể hiện những báo động trên thanh trạng thái.

- **Activity Manager** quản lý vòng đời của các ứng dụng và cung cấp một backstack cho phép chuyển đổi giữa các ứng dụng.

#### Libraries

Android bao gồm một bộ thư việc C/C++ được dùng bởi những bộ phận của hệ thống Android. Những khả năng này được bộ lộ thông qua Android application framework. Một số những thư viện cốt lõi được liệt kê dưới đây:

- **System C library**: một implementation dẫn xuất từ BSD của thư viện hệ thống chuẩn C (libc), được điều chỉnh cho những thiết bị nhúng dựa trên Linux.

- **Media libraries**: dựa trên OpenCORE của PacketVideo, những thư viện này hỗ trợ playback vào thâu những định dạng

- **Surface Manager**: quản lý sự truy cập tới các hệ thống con và kết hợp những lớp đồ họa 2D và 3D từ nhiều ứng dụng.

- **LibWebCore**: một web browser engine có thể vận hành cả Android browser và những WebView nhúng.

- **SGL**: một graphics engine 2D nằm bên dưới.

- **3D libraries**: một implementation dựa trên OpenGL ES 1.0 APIs; những thư viện này sử dụng hardware 3D acceleration (khi có thể) hoặc những bộ quét 3D được tối ưu.

- **FreeType**: cách vẽ font vector và font bitmap.

- **SQLite**: một lightweight relational database engine mạnh mẽ tồn tại cho tất cả ứng dụng.

#### Android Runtime

Android gồm một bộ thư viện cốt lõi nhằm cung cấp hầu hết những tính năng có sẵn trên những thư viện cốt lõi của ngôn ngữ lập trình Java.

Mỗi ứng dụng Android chạy trên những process của riêng nó trên những thể hiện của Dalvik virtual machien. Dalvik được viết để một thiết bị có thể chạy nhiều VMs một cách hiệu quả. Dalvik VM thực thi file có định dạng Dalvik Executable (.dex) nhằm tối ưu cho những hoạt động đòi hỏi bộ nhớ. VM này sẽ được đăng ký, chuyển đổi những class file được biên dịch bởi Java compiler thành những file có định dạng .dex, và cuối cùng thực thi chúng. Dalvik VM trông cậy vào Linux kernel vì những tính năng như đa luồng hay quản lý bộ nhớ cấp thấp.

#### Linux kernel

Android trông cậy vào những dịch vụ của Linux phiên bản 2.6 như security, quản lý bộ nhớ, quản lý tiến trình, network stack, driver model. Kernel này cũng hành động như một lớp trừu tượng giữa phần cứng và phần còn lại của software stack.

### Kiến thức cơ bản

Ứng dụng Android được viết bằng ngôn ngữ lập trình Java. Android SDK biên dịch code cùng với dữ liệu và resource files, đặt vào một gói gọi là Android package, một file nén với đuôi “.apk”. Tất cả code trong file apk được xem như một ứng dụng và dùng để cài vào thiết bị. Một khi được cài trên thiết bị, mỗi ứng dụng Android chạy như một security sandbox:

- Hệ điều hành Android là một hệ thống multi-user, nơi mà mỗi ứng dụng là một user khác nhau.

- Mặc định, hệ thống sẽ gán mỗi ứng dụng một Linux user ID (ID chỉ được sử dụng bởi hệ thống và ứng dụng sẽ không biết). Hệ thống cài đặt quyền cho tất cả files trong ứng dụng để cho chỉ duy nhất user có ID được gán cho ứng dụng mới có quyền truy xuất chúng.

- Mỗi process có máy ảo riêng của chúng, nên code của một ứng dụng sẽ chạy tách biệt với ứng dụng khác.

- Mặc định, mỗi ứng dụng chạy trong một Linux process. Android sẽ khởi động process khi bất kỳ bộ phận nào cần được thực thi, sau đó sẽ shut down process đó khi không cần nó nữa, hoặc khi hệ thống phải khôi phục bộ nhớ để cấp phát cho ứng dụng khác.

Như đã thấy, hệ thống Android thực hiện theo nguyên tắc đặc quyền tối thiểu (principle of least privilege). Đó là, mỗi ứng dụng mặc định sẽ truy xuất chỉ những bộ phận mà nó yêu cầu. Điều này tạo ra một môi trường an toàn, ngăn một ứng dụng truy xuất vào những phần của hệ thống khi không được phép.

Tuy nhiên, có nhiều cách để một ứng dụng chia sẽ dữ liệu với ứng dụng khác và một ứng dụng truy xuất những dịch vụ của hệ thống.

- Có thể sắp xếp để hai ứng dụng chia sẻ cùng Linux user ID để chúng có thể truy xuất file của nhau. Để bảo toàn tài nguyên hệ thống, những ứng dụng với cùng user ID có thể chạy trên cùng Linux process và chia sẻ cùng VM (những ứng dụng phải được sign với cùng certificate).

- Một ứng dụng có thể yêu cầu quyền để truy xuất dữ liệu của thiết bị như user’s contacts, SMS messages, SD card, camera, Bluetooth, và hơn thế nữa. Người dùng phải cài đặt quyền cho ứng dụng ở thời điểm cài đặt.

Những điều trên đã mô tả khái quát làm thế nào mà một ứng dụng Android tồn tại được trong hệ thống.

#### Application components

Application components được xem như những “trụ cột” chính để xây dựng nên một ứng dụng Android. Mỗi component là một ngõ để hệ thống có thể vào ứng dụng. Không phải tất cả component đều là ngõ chính cho người dùng và một số phụ thuộc vào nhau, nhưng mỗi cái tồn tại và đóng vai trò riêng của nó, giúp định hình hành vi tổng thể của một ứng dụng.

Có 4 loại application component. Mỗi loại phục vụ một mục đích khác nhau và có vòng đời khác nhau.

#### Activities

Một activity đại diên cho một màn hình tương tác với người dùng. Ví dụ, một ứng dụng email có thể có một activity thể hiện một danh sách email, một activity khác để soạn email, và một cái khác để đọc email. Mặc dùng các activity hoạt động cùng nhau để định hình một ứng dụng, nhưng mỗi cái độc lập với nhau. Với cách này, một ứng dụng khác có thể chạy bất kỳ activity nào trong số này (nếu ứng dụng email cho phép điều đó). Ví dụ, một camera application có thể chạy activity trong ứng dụng email để soạn mail và chia sẻ hình ảnh.

#### Services

Một service là một component chạy bên dưới để thực thi một thao tác lâu và kéo dài, hoặc thực hiện một remote process. Một service không cung cấp giao diện. Ví dụ một service chơi nhạc ở bên dưới khi người dùng đang ở trong một ưng dụng khác, hoặc nó có thể lấy dữ liệu từ network mà không block tương tác của người dùng với một activity. Một component khác như một activity có thể khởi động service và để nó chạy hoặc gắn vào để tương tác với nó.

#### Content providers

Một content provider quản lý tập hợp dữ liệu có thể chia sẻ giữa các ứng dụng. Bạn có thể lưu dữ liệu trên một file system, một SQLite database, tên web, hay trên bất kỳ persistent storage location nào mà bạn có thể truy xuất. Với content provider, những ứng dụng khác có thể truy vấn hoặc thay đổi dữ liệu (nếu content provider cho phép). Ví dụ, hệ thống Android với những quyền phù hợp có thể truy vấn những phần của content provider như contacts để đọc và ghi thông tin về một người nào đó. Content provider cũng hữu dụng khi đọc và ghi dữ liệu private với ứng dụng và không được chi sẻ.

#### Broadcast receivers:

Broadcast receiver là một component trả lời những broadcast announcements rộng khắp ứng dụng. Nhiều broadcast bắt nguồn từ hệ thống như broadcast thông báo rằng màn hình bị tắt, pin yếu, hay ảnh được chụp. Các ứng dụng có thể tạo broadcast, ví dụ để cho ứng dụng biết một số dữ liệu được download về thiết bị và có sẵn để sử dụng. Mặc dù broadcasts receiver không hiện một giao diện nhưng chúng có thể tạo notification để hiển thị lên status bar, thông báo cho người dùng khi một broadcast xảy ra. Nói chung, broadcast receiver chỉ là một “gateway” đối với những component khác và làm những cô việc nhỏ. Ví dụ như khởi tạo một dịch vụ để thực hiện một số công việc dựa trên sự kiện.

Một khía cạnh duy nhất của thiết kế hệ thống Android là bất kỳ ứng dụng nào cũng có thể start một component của ứng dụng khác. Ví dụ, nếu bạn muốn user chụp ảnh với camera của thiết bị, ứng dụng của bạn có thể dùng chức năng chụp ảnh của ứng dụng camera để làm điều đó, không cần phải kết hợp hay link đến code của ứng dụng camera. Đơn giản là start một activity trong camera application để chụp ảnh, khi hoàn tất thì ảnh sẽ đc trả về cho ứng dụng của bạn.

Khi hệ thống start component, nó sẽ start process dành cho ứng dụng đó (nếu nó chưa chạy) và thể hiện những lớp cần thiết cho component này. Ví dụ, nếu ứng dụng của bạn start activity trong ứng dụng camera để chụp ảnh, activity đó sẽ chạy trong process thuộc về ứng dụng camera, không phải là process trong ứng dụng của bạn. Do đó, không giống những ứng dụng trên hầu hết hệ thống, ứng dụng Android không có điểm bắt đầu chính (nghĩa là không có hàm main())

Vì hệ thống chạy mỗi ứng dụng trên một process riêng biệt với quyền truy xuất file giới hạn với ứng dụng khác nên ứng dụng của bạn không thể trực tiếp kích hoạt một component từ ứng dụng khác. Tuy nhiên, hệ thống Android có thể, để kích hoạt một component trong ứng dụng khác, bạn phải gởi thông điệp cho hệ thống đó để chỉ định mục đích (intent) để start component đó. Thệ thống sau đó kích hoạt component đó cho bạn.

#### Activating components

Ba trong bốn loại component là activity, service và broadcast receiver được kích hoạt bởi một thông điệp bất đồng bộ (asynchronous message) được gọi là một intent. Intent cột những component với nhau ở thời gian thực thi (bạn có thể nghĩ về chúng như những sứ giả yêu cầu một hành động từ những component), cho dù component có thuộc ứng dụng của bạn hay không.

Một intent được tạo với một Intent object, cái định nghĩa một thông điệp để kích hoạt một component chỉ định hay một kiểu component.

Đối với activity và service, một intent định nghĩa hành động thực hiện (ví dụ xem hoặc gởi một cái gì đó) và có thể chỉ định URI của dữ liệu. Ví dụ, một intent có thể mang một yêu cầu đến một activity để hiển thị một bức ảnh hoặc mở một trang web. Trong một số trường hợp, bạn có thể start một activity để nhận kết quả. Activity cũng có thể trả về kết quả trong một Intent (ví dụ, bạn có thể tạo ra một intent để người dùng chọn một contact và trả về cho bạn, intent được trả về sẽ bao gồm URI trỏ đến contact đó).

Đối với broadcast receiver, intent đơn giản là định nghĩa thông báo được gởi đi. Ví dụ một broadcast để báo hiệu pin yếu bao gồm một chuỗi như “Pin yếu. Làm ơn sạc pin”.

Riêng content provider không được kích hoạt bởi intent. Thay vào đó, nó được kích hoạt khi nhận được yêu cầu từ ContentResolver. Content resolver sẽ quản lý tất cả transaction trực tiếp với content provider để component đang thực hiện transaction với provider này chỉ cần gọi phương thức trên ContentResolver. Điều này sẽ tạo ra một layer trừu tượng giữa content provider và component yêu cầu thông tin.

#### The manifest file

Trước khi hệ thống Android có thể start component của ứng dụng, hệ thống phải biết component đang tồn tại bằng cách đọc AndroidManifest.xml (manifest file). Ứng dụng của bạn phải khai báo tất cả component trong file này.

Ngoài việc khai báo component, manifest file làm một số việc như sau:

- Nhận dạng user permission mà ứng dụng yêu cầu, ví dụ như Internet access hay quyền đọc đối với user’s contacts.

- Khai báo API level tối thiểu được yêu cầu bởi ứng dụng, dựa trên APIs mà ứng dụng dùng.

- Khai báo các đặc tính software và hardware được yêu cầu bởi ứng dụng, ví dụ như camera, bluetooth hay multitouch.

- Những API libraries mà ứng dụng cần được liên kết đến, ví dụ như Google Maps library.

- Và nhiều thứ khác nữa.

#### Application resources

Một ứng dụng Android bao gồm nhiều thứ, không chỉ có code, nó yêu cầu những tài nguyên tách biệt với source code, ví dụ như ảnh, audio file hay bất cứ thứ gì liên quan tới việc hiển thị của ứng dụng. Ví dụ, bạn sẽ định nghĩa những hình động (animation), menus, styles, colors và layout của giao diện activity bằng XML files. Dùng resources sẽ giúp dễ dàng hơn trong việc cập nhật giao diện mà không phải thay đổi code. Dùng một bộ resources thay thế cho nhau sẽ cho phép bạn tối ưu ứng dụng đối với những thiết lập thiết bị khác nhau (ví dụ như ngôn ngữ và kích thước màn hình khác nhau).

Đối với mỗi resource bạn include trong project, SDK build tools sẽ định nghĩa một ID duy nhất để có thể tham chiếu đến resources đó trong code ứng dụng hoặc từ những resource khắc được định nghĩa trong XML. Ví dụ, nếu ứng dụng chứa một file ảnh tên là logo.png (lưu trong thư mục res/drawable/), SDK tools sẽ sinh ra resource ID là R.drawable.log để bạn có thể dùng để tham chiếu đến bức ảnh và chèn nó vào giao diện.

Một trong số những khía cạnh quan trọng trong việc cung cấp resources tách biệt với code là bạn có thể cung cấp một resource thay thế cho những cấu hình thiết bị khác nhau. Ví dụ, có thể tạo ra những file chuỗi với những ngôn ngữ khác nhau để thay thế cho nhau. Sau đó dựa trên ngôn ngữ được chọn trong hệ thống, Android sẽ áp dụng ngôn ngữ phù hợp cho UI.

Android hỗ trợ nhiều qualifiers cho những resources thay thế. Qualifier là một chuỗi ngắn được gắn trong tên của thư mục resources để xác định resource nào nên được dùng để phù hợp với cấu hình của thiết bị. Ví dụ, bạn thường tạo ra những layout khác nhau cho những activity khác nhau phụ thuộc vào orientation của màn hình và kích thước. Ví dụ khi màn hình ở chết độ chân dung (portrait orientation), bạn muốn layout sẽ có button theo chiều dọc, trong khi màn hình trong chế độ phong cảnh (landscape orientation) thì button sẽ theo chiều ngang. Để thay đổi layout phụ thuộc vào orientation, bạn có thể định nghĩa những layout khác nhau và áp dụng qualifier phù hợp cho mỗi tên thư mục. Sau đó, hệ thống sẽ tự động áp dụng layout phù hợp phụ thuộc vào orientation của thiết bị.

### Giới thiệu SQLite.

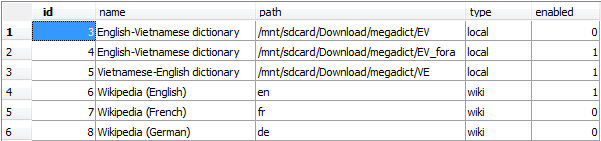
SQLite được gọi là “in-process library”, với các tính chất: tự chứa (self-contained), không có server (serverless), không cấu hình (zero-configuration), T-SQL database engine.

SQLite là dạng cơ sở dữ liệu nhúng. Không giống như hầu hết các cơ sở dữ liệu khác, SQLite không có một server chạy riêng. SQLite đọc và ghi trực tiếp vào file. Một cơ sở dữ liệu với bảng, indices, triggers, views được chứa chỉ trong một file duy nhất. Định dạng của SQLite là không phụ thuộc nền (cross-platform), ta có thể copy database file giữa hệ thống 32-bit và 64-bit. Những tính chất này làm cho SQLite trở thành sự lựa chọn thông dụng như một định dạng file ứng dụng (application file format).

SQLite có thể chạy trong những môi trường ít bộ nhớ: bộ nhớ stack là 4KiB với heap rất nhỏ là 100KiB. Điều này làm cho SQLite trở thành database engine thông dụng trên những thiết bị di động như cellphone, PDAs và MP3.

### SQLite trong MegaDict

MegaDict dùng SQLite ở mức độ đơn giản, đó là chỉ tạo database với 1 bảng duy nhất.



Hình ‑ – Cơ sở dữ liệu của MegaDict

+ Name là dictionary name.

+ Path là đường dẫn lưu dictionary trên SD card.

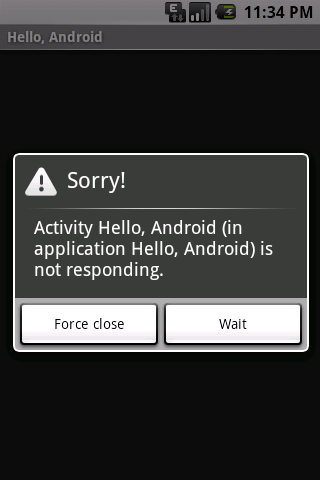
+ Type là loại từ điển. Hiện tại MegaDict gồm 2 loại: local và wiki.

+ Enabled xác định từ điển đó có đang được dùng hay không

Database được quản lý bởi một lớp được implement Singleton pattern, chỉ được open khi được dùng đến, và sẽ tự động được close khi ứng dụng bị hủy hoàn toàn.

### Thiết kế khả năng đáp lại của ứng dụng (Designing for responsiveness).

Trong Android, có một số ứng dụng phải tiêu tốn một khoảng thời gian để thực thi. Khi người dùng tương tác với ứng dụng, có thể họ sẽ phải đợi để nhận được kết quả. Vì lẽ đó, hệ thống của Android sẽ bật một hộp thoại gọi là Application Not Responding (ANR). Người dùng có thể chọn tắt ứng dụng (Force close) hoặc tiếp tục đợi quá trình kết thúc (Wait). Người dùng sẽ cảm thấy phiền toái khi cứ phải trả lời hộp thoại này. Vì vậy, trách nhiệm của người lập trình là phải tránh hiện tượng ANR khi có thể.



Nói cách khác là hệ thống chỉ hiện ANR khi ứng dụng không thể trả lời user input.

+ Ví dụ, nếu ứng dụng block một số thao tác I/O thì ứng dụng chủ không thể xử lý những sự kiện nhập được nữa. Lúc đó hệ thống kết luận rằng ứng dụng bị treo hoặc tạm ngưng (frozen) và hiển thị hộp thoại ANR.

+ Một ví dụ khác là nếu ứng dụng tốn quá nhiều thời gian cho một thao tác đòi hỏi bộ nhớ, ví dụ như tính toán nước đi tiếp theo trong một trò chơi, thì ứng dụng cũng sẽ tạm ngưng.

Trong cả hai trường hợp trên, một cách giải quyết được khuyên dùng là tạo ra luồng con (child thread) và thao tác ở đó. Điều này giúp cho main thread (luồng ứng dụng chính điều khiển vòng lặp sự kiện của UI) tiếp tục chạy và ngăn ANR hiển thị.

Điều gì đã gây ra ANR?

Android sẽ hiện hộp thoại ANR nếu như một trong hai điều sau xảy ra:

+ Không có trả lời đáp lại đối với user input trong 5 giây. (ví dụ: nhấn bàn phím, chạm màn hình)

+ Một BroadcastReceiver chưa hoàn tất thực thi trong 10 giây.

Làm sao tránh ANR?

Bình thường ứng dụng Android chạy trên một luồng riêng biệt được gọi là main thread. Điều này có nghĩa là bất kỳ công việc nào có thời gian lâu trong main thread cũng có thể khiến ANR hiển thị bởi vì nó sẽ không cho phép người dùng tương tác vào ứng dụng nữa.

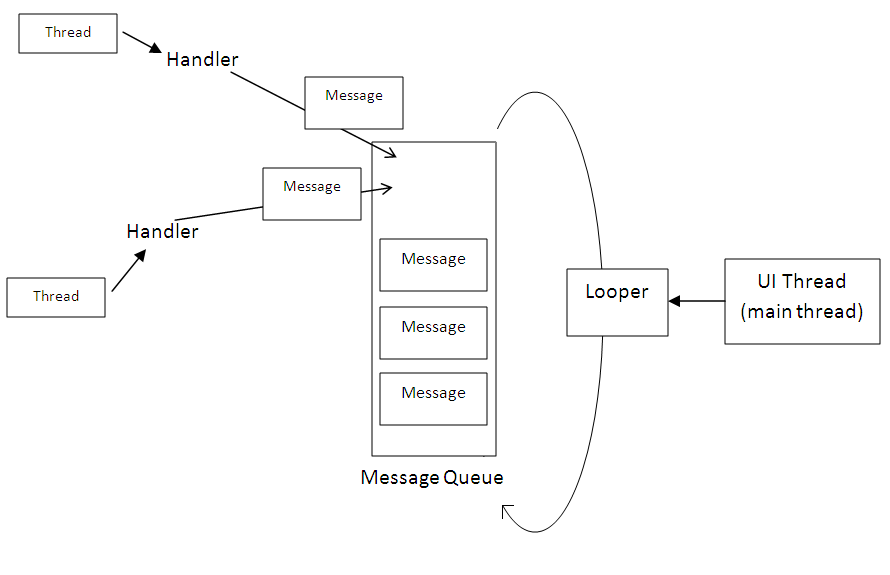
Do đó, lời khuyên là chạy càng ít công việc trong main thread càng tốt. Giải pháp là thực thi những công việc tốn thời gian (long time method) trong những luồng con, còn được gọi là những thao tác chạy ngầm (background operations). Ví dụ như: truy xuất network hay truy xuất database, những công việc tính toán phức tạp hay xử lý bitmap. Điều này giúp cho main thread không bị block và người dùng vẫn có thể tương tác với UI.

Những thao tác chạy ngầm (background operations) phải cho phép update UI trên UI thread để khi hoàn tất một child thread thì UI thread có thể được cập nhật dễ dàng. Android hỗ trợ 2 cách để thực hiện background operation:

+ Dùng Handler.

+ Dùng AsyncTask.

1) Dùng Handler:



Hình ‑ – Dùng Handler

Handler trong Android áp dụng Pipeline Thread Design pattern. Ý tưởng là có 3 bộ phận quan trọng là Looper, Handler và Queue of tasks (hay còn gọi là message queue).

+ Công việc của Queue of task là giữ những task trong một hàng đợi để được thực thi sau đó. Những luồng khác có thể push task vào Queue of tasks ở bất kỳ thời điểm nào.

+ Công việc của Looper là lặp qua Queue of tasks để lấy từng task ra và thực thi. (task còn được gọi là message)

+ Handler class có một giao diện thread-safe, có thể được truyền cho những thread khác. Những thread khác sẽ dùng thao tác postMessage của Handler để post task vào Queue of task của pipeline thread. Handler cũng được gắn (attached) vào UI thread để có thể update UI trên UI thread.

Vì Activity class của Android đã implement một Looper bên trong nên khi khai báo Handler trong Activity, Handler đó sẽ tự được được cột (bind) vào Looper đó.

2) Dùng AsyncTask:

Vì cách dùng Handler hơi phức tạp và đòi hỏi phải implement bằng tay khá nhiều chi tiết nên Android hỗ trợ một giao diện dễ sử dụng hơn, đó là AsyncTask. Thực ra AsyncTask cũng dự trên Pipeline thread pattern được đề cập ở trên nhưng nó cung cấp một giao diện dùng ngay.

Để dùng AsyncTask, ta extend nó và implement 4 phương thức: onPreExecute(), doInBackground(), onProgressUpdate() và onPostExecute().

Tóm tắt ý nghĩa của những phương thức này như sau:

+ onPreExecute được thực thi trên UI thread sau khi task này được thực thi. Phương thức này dùng để setup một số thứ trước khi thực thi task.

+ doInBackground(Params …) được gọi trên background thread sau onPreExecute. Bước này dùng để thực thi những thao tác tính toán thời gian dài. Kết quả tính toán được trả về và được truyền cho onPostExecute như tham số.

+ onProgressUpdate(Progress …) được gọi trên UI thread sau khi một cú gọi đến publishProgress(Progress …). Thời gian thực thi là không xác định. Phương thức này được dùng để cập nhật UI khi việc tính toán ngầm trong doInBackground() vẫn đang được thực hiện.

+onPostExecute(Result) được gọi trên UI thread khi việc tính toán đã hoàn tất. Tham số của nó là giá trị được trả về từ doInBackground().

Với 2 cách làm trên, ta sẽ tránh được hiện tượng ANR.

MegaDict áp dụng cách thứ hai để thực hiện tìm kiếm một từ trong từ điển. Thao tác onPreExcecute hiển thị ProgressBar, hiển thị nó; doInBackground làm công việc tìm kiếm và onPostExcecute hiển thị nội dung trên WebView, đồng thời ẩn ProgressBar.

### Vòng đời của Activity:

Lập trình Android chủ yếu xoay quanh sự vận hành của Activity. Activity là một màn hình cung cấp giao diện tương tác cho người sử dụng. Một ứng dụng có thể gồm nhiều activity nhưng chỉ có duy nhất một main Activity. Main Activity xuất hiện khi người dùng khởi động ứng dựng.

Mỗi lần một activity start thì activity hiện hành sẽ stop, nhưng hệ thống vẫn bảo quản activity đó trong một stack, gọi là “back stack”.

Khi một activity start, nó được push vào stack và người dùng có thể focus trên đó. Khi một activity stop do một activity khác start, nó được thông báo về sự thay đổi trạng thái này thông bao những hàm callback.

Có nhiều hàm callback mà một activity có thể nhận và người dùng sẽ có cơ hội thực hiện những việc cần thiết phù hợp với trạng thái đó. Ví dụ, khi activity stop, bạn có thể giải phóng những đối tượng lớn như network hay database connection. Khi activity resume, bạn có thể đạt lại những resource cần thiết và resume lại hành động đã bị ngắt. Chuyển trạng thái là một phần quan trọng trong vòng đời của Activity.

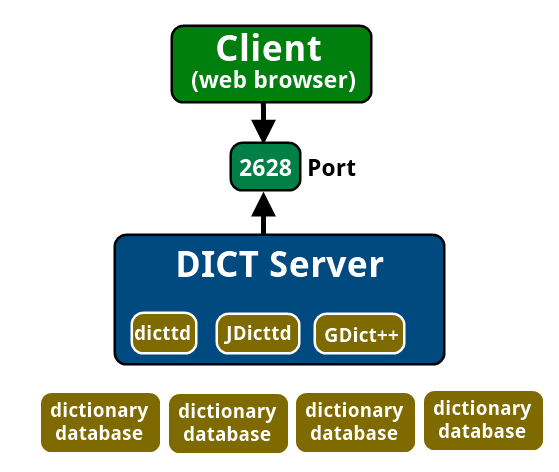


Hình ‑ – Vòng đời của Activity

## DICT Dictionary Format

### Giao thức DICT (DICT Protocol)

DICT là một giao thức từ điển mạng (dictionary network protocol) được phát triển bởi DICT Development Group. Mục đích chính của giao thức này là tìm một giải pháp mới để thay thế cho giao thức Webster, cho phép người sử dụng có thể truy xuất nhiều từ điển trong một lần sử dụng. Dict server và client sử dụng port 2628 để giao tiếp với nhau.



Hình ‑ - Kiến trúc của DICT Protocol

Toàn bộ mô tả giao thức (specification) có thể đọc tại <http://tools.ietf.org/html/rfc2229>.

Mục đich chính của DICT là tạo ra một hệ thống từ điển trực tuyến, sử dụng thông qua Internet, trong đó lượng database miễn phí được chia sẻ từ khắp nơi.

Có rất nhiều cài đặt DICT server bằng nhiều ngôn ngữ khác nhau như:

* *Dicttd* (ứng dụng server chuẩn được phát triển bởi DICT Development Group).
* *GNU Dico*: một server được phát triển bởi GNU, hỗ trợ các định dạng từ điển khác ngoài DICT format. Mỗi một format được định nghĩa là một loadable module và có thể viết bằng C, Python hay Guile.
* *JDicttd*: được phát triển bởi Ho Ngoc Duc, viết hoàn toàn bằng Java. Hỗ trợ định dạng nén và dạng file chuẩn của DICT.
* *DictD++*: một tân binh server được viết bằng C++ (sử dụng STL, Boost và một vài thư viện khác), chạy trên FreeBSD, Linux Mandrake, Windows 2K/XP.

### Định dạng từ điển của DICT Protocol

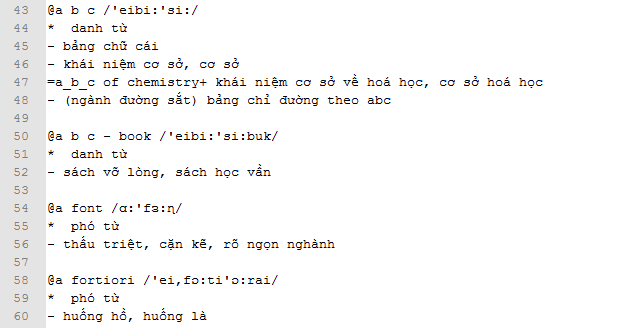
Cơ sở dữ liệu của DICT Protocol được phân phố dưới dạng riêng rẽ. Mỗi một cơ sở dữ liệu bao gồm hai file định dạng plain text: một được gọi là ***flat text*** (\*.dict) và ***index file*** (\*.index).

Cả 2 file đều là file thuần text và được thiết kể để có thể đọc được bằng bất cứ trình text editor nào (human-readable).

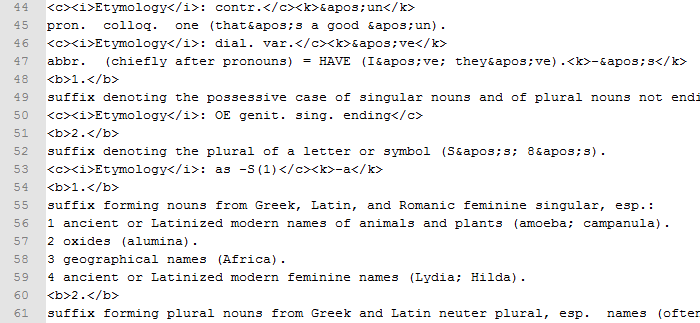
**Flat text (Dictionary file)**

* Đây là tập tin chứa nội dung chính định nghĩa của các từ mà từ điển hỗ trợ. Nội dung có thể là thuần text hay chứa HTML tag.
* Nội dung của file được tạo ra bởi một công cụ được cung cấp sẵn bởi DICT Development Group được gọi là dictfmt. Cấu trúc của flat file được xây dựng sao cho client có thể truy xuất ngẫu nhiên (random access) nhằm giảm thời gian load file và tối ưu khả năng truy xuất đa luồng từ server.

Một số ví dụ:



Hình ‑ - Nội dung thuần text của flat file



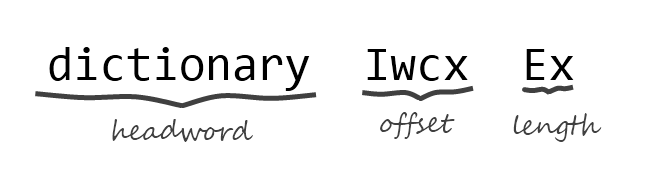
Hình ‑ - Nội dung file flat text chứa các tag đặc biệt

**Index File**

* Index file là file chứa tất cả các từ khóa mà từ điển hỗ trợ. DICT Server sẽ tìm kiếm từ khóa trên file index, nếu có, nội dung được mã hóa trong file index sẽ được chuyển đổi về đúng định dạng và tiến hành lấy dữ liệu từ flat text.
* Một file index bao gồm các dòng, mỗi dòng là một bộ định nghĩa (record) của từ điển được phân cách bởi ký tự TAB:

1. Headword: từ khóa.
2. Offset: vị trí bắt đầu của dữ liệu trong flat text.
3. Length: độ dài của dữ liệu bắt đầu từ offset.

Ví dụ:



Hình ‑ - Nội dung một record trong index file

* Nội dung của Offset và Length là số và được mã hóa bởi giải thuật Base64 Encoding. Mục đích nhắm làm giảm số lượng ký tự sử dụng trong việc thể hiện số.

**Mã hóa Base64 trong DICT Protocol format.**

Mã hóa Base64 là một dạng mã hóa sử dụng 64 ký tự ASCII để chuyển binary data (số, byte,..) sang dạng có thể biểu diễn trên text editor (sử dụng ASCII),

Dưới đây là bảng các ký tự ASCII được sử dụng trong Base64 encoding và giá trị của nó so với binary data.



Hình ‑ - Bảng các ký tự được sử dụng trong mã hóa Base64

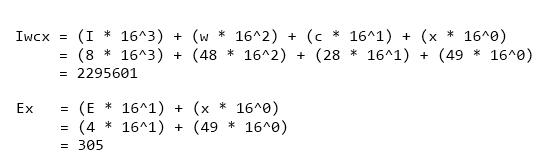
Trong DICT Protocol, các giá trị offset và length của index record được mã hóa bởi Base64 nhằm tiết kiệm số lượng ký tự dùng để thể hiện số. Để có thể chuyện ngược dữ liệu đã được mã hóa, tao chỉ cần thực hiện giải thuật tương tự như chuyển Base10 sang Base2.

Ví dụ:

Nếu record có nội dung

dictionary Iwcx Ex

thì:



Có nghĩa là từ nội dung của từ dictionary sẽ bắt đầu từ vị trí 2295601 và có độ dài (tính theo byte) 305.

### Sử dụng trực tiếp DICT Protocol database

Do tính chất của DICT Protocol chủ yếu phục vụ cho ứng dụng web client, cho nên sử dụng DICT Protocol trong đề án thực sự không hiệu quả. Nhóm thực hiện đề án quyết định tự implement cách đọc riêng cho megadict.

Nghĩa là vẫn sử dụng định dạng từ điển của DICT Protocol, nhưng phương pháp đọc và xử lý tìm kiếm và đọc dữ liệu được implement mới hoàn toàn và không giống với phương pháp được sử dụng ở DICT Protocol.

## Java NIO



### New I/O

Input/Output (I/O) là một trong những phần căn bản mà bất cứ một người lập trình máy tính nào cũng phải một lần sử dụng khi phát triển một phần mềm. Hiện thực quá trình I/O lúc nào cũng là một thử thách lớn, nhất là tốc độ thực hiện vì tốc độ của I/O phụ thuộc vào nhiều yếu tố: nền tảng hệ điều hành, tốc độ thiết bị lưu trữ,…

Trước version 1.4, Java cung cấp một gói thư viện IO cho người sử dụng là java.io, nhưng nó không đáp ứng được yêu cầu ngày càng cao của người sử dụng. Do đó, Sun Microsystem đã nghiên cứu và cung cấp gói thư viện IO mới, New IO – java.nio, nhằm mục đích cung cấp cho người sử dụng một thư viện có hiệu suất hoạt động cao, portable và scalable.

Java NIO cung cấp các chức năng mở rộng sau:

Buffer đối với primitive types.

* Bộ encoders/decoders cho character-sets.
* Hỗ trợ regular expression
* Cung cấp một abstraction mới, Channel
* Interface mới hỗ trợ locking và memory mapping
* Multiplexed, non-blocking I/O

### Input/Output với hiệu suất sao

Với các version trước Java SDK 1.4, lập trình viên chỉ có một lựa chọn duy nhất cho I/O là sử dụng stream. Có 2 dạng stream cơ bản phục vụ cho tất cả các tác vụ IO của Java tại thời điểm đó: *byte stream* và *character stream*. Java IO cung cấp API để người sử dụng đọc dữ liệu từ stream vào array hay ghi dữ liệu từ array vào stream.

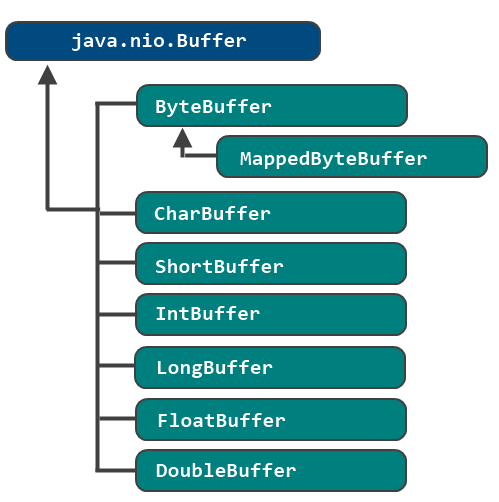
Điểm yếu của Java IO là sử dụng cơ chế blocking, nghĩa là stream sẽ bị block cho tới khi native IO thực hiện xong. Phương thức read() sẽ không return cho tới khi tất cả dữ liệu được đọc vào stream và write() chỉ return khi tất cả dữ liệu được ghi vào stream .Cơ chế này gây ra hiện tượng thắt cổ chai khi thực hiện I/O với hiệu năng cao.

Ngoài việc sử dụng cơ chế blocking, Java IO để lập trình viên phải sử dụng tới raw bytes cũng là một bất tiện trong việc thực hiện IO. Để tăng tốc IO, lập trình viên thường sử dụng một primitive array và truyền vào dữ liệu trực tiếp từ stream. Người sử dụng phải tự đọc raw bytes và kiểm soát EOF (end-of-file). Để tránh việc sử dụng buffering trực tiếp như vậy, người sử dụng được cung cấp một loạt các class implement sẵn buffering như BufferInputStream, BufferReader, BufferWriter,..và sử dụng Decorater Pattern, kết nối một stream vào các lớp buffer này để thực hiện IO. Việc sử dụng này gây bất tiện và rắc rối.

Với những bất lợi đó, từ version 1.4, Java SDK cung cấp gói java.nio với những chức năng và kiến trúc mới nhằm khắc phục các thiếu sót và gói java.io gặp phải. Java NIO cung cấp API tiện lợi và dễ sử dụng hơn, cung cấp các concept mới: Buffer và Channel.

### Buffer

Một Buffer đơn giản chỉ là một container mà bên trong là một chuỗi có giới hạn các primitive type của Java, đơn giản mà nói một Buffer chỉ là một wrapper của một array primitive type. Java NIO cung cấp mỗi một primitive type một loại buffer.



Hình ‑ – Buffer và các subclass

Lớp Buffer là một abstract class cung cấp hầu hết các thao tác mặc định mà một Buffer cần có. Buffer có 7 subclass, tương ứng với 7 primitive type của Java: ByteBuffer, CharacterBuffer, ShortBuffer, IntBuffer, LongBuffer, FloatBuffer, DoubleBuffer.

Một Buffer có 3 property chính quan trọng: *capacity*, *limit* và *position*. Các property này được set lúc khởi tạo buffer và trong lúc sử dụng.

Buffer được khởi tạo bởi 2 cách: sử dụng phương thức allocate hay allocateDirect, vốn nhận một tham số là một integer, ấn định *capacity* của Buffer. Điểm khác biện giữa 2 phương thức này là allocateDirect sẽ khởi tạo một buffer nằm bên ngoài heap của JVM, giúp bỏ bới thao tác copy dữ liệu giữa JVM buffer và OS buffer, kết quả là tăng tốc đáng kể tác vụ I/O. Tuy nhiên, việc này cũng có 1 tradeoff, đó là quy trình khởi tạo buffer sẽ tốn nhiều thời gian và công sức hơn. Tuy nhiên yếu điểm này cũng có thể giảm nếu sử dụng một vài practice đúng cách để tối ưu hóa.

// Khởi tạo một buffer có capacity là 5000 bytes

ByteBuffer heapBuffer = ByteBuffer.*allocate*(5000);

// Khởi tạo một DIRECT buffer có capacity là 5000 bytes;

ByteBuffer directBuffer = ByteBuffer.*allocateDirect*(5000);

// Sử dụng buffer

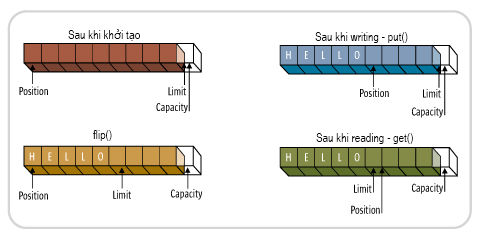
String text = "Hello";

**byte**[] textInBytes = text.getBytes();

heapBuffer.put(textInBytes);

Buffer cung cấp một số phương thức để sử dụng đơn giản:

* Phương thức flip: chuẩn bị buffer cho một chuỗi thao tác channel-write hay relative get, phương thức này sẽ set giá trị của limit về position hiện tại và position trở về 0.
* Phương thức get: lấy dữ liệu từ buffer để sử dụng. Có 2 loại: *relative get* lấy dữ liệu từ position hiện tại, và *bulk get*: lấy một lượng dữ liệu từ position hiện tại vào một mảng array.
* Phương thức put: bỏ dữ liệu vào buffer. Cũng có 2 loại giống như get.
* Phương thức rewind: chuẩn bị buffer để thực hiện phương thức get sau khi sử dụng phương thức put. Phương thức này giữ nguyên limit và chuyển position về 0.
* Phương thức clear: chuẩn bị buffer để thực hiện các thao tác channel-read hay relative put. Dời position về 0 và limit tới capacity.
* Phương thức reset: reset buffer. Dời position về 0, limit tới capacity, và bỏ giá trị của mark.

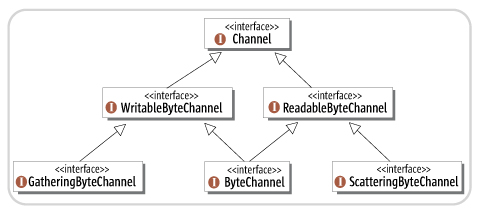


Hình ‑ – Giá trị của buffer property khi thực hiện các phương thức

### Channels

Channel là một khái niệm mới được đưa vào Java NIO. Một channel thể hiện một kết nối được mở tới một thực thể có thể thực hiện một hay nhiều tác vụ I/O như thiết bị phần cứng, file, network socket, hay program component. Thí dụ với FileChannel, SocketChannel thể hiện một kênh nối tới một tập tin hay một socket.

Điểm khác biệt lớn nhất giữa stream và channel là với stream, việc thực hiện thao tác read hay write phải tách biệt nhau nhưng với channel thì có thể thực hiện đồng thời cả 2 việc. Điểm khác biệt nữa là trong khi stream hoạt động dựa trên raw bytes thì channel dựa trên buffer.



Hình ‑ – Hierachy của Channel

Interface Channel và các subinterface được định nghĩa trong gói java.nio.channels. Ba interface: ByteChannel, ReadableByteChannel, WritableByteChannel có thể đoán được chức năng của nó từ tên gọi. ScatteringByteChannel và GatheringByteChannel cung cấp khả năng đọc và ghi bytes từ nhiều buffer khác nhau chỉ bằng một lệnh thực thi.

### FileChannel và Memory Mapping

FileChannel tất nhiên là channel thường được sử dụng nhất. Lập trình viên có thể sử dụng FileChannel để đọc, ghi, map, thay đổi nội dung file. Mapping file vớ FileChannel được nâng cấp lên một mức hoàn toàn mới.

Với Java NIO, hoàn toàn có thể map một phần hay toàn bộ file trực tiếp vào bộ nhớ. Bất cứ thay đổi nào xày ra với dữ liệu được map sẽ được cập nhất tới file thật sự. Chức năng này thực sự là một sự bổ sung thiết yếu bởi trước đây để thực hiện được, lập trình viên phải sử dụng native code.

Thực hiện map file chỉ đơn giản như sau:

File largeFile = **new** File("VeryBigLargeFile.dat");

RandomAccessFile raf = **new** RandomAccessFile(largeFile, "rw");

FileChannel fileChannel = raf.getChannel();

MappedByteBuffer mappedByteBuffer = fileChannel.map(MapMode.READ\_WRITE, 0, largeFile.length());

Phương thức map của FileChannel sẽ trả về một MappedByteBuffer, được thiết kế đặc biệt cho chức năng mapping. Ưu điểm chính của việc map file là nội dung của file sẽ được map và sử dụng như là virtual memory của OS. Nếu một phần nào đó của file bị thay đổi, chỉ có phần đó được load vào bộ nhớ vật lý. Với cơ chế này, thích hợp cho việc làm việc với file có dung lượng lớn (có thể là gigabyte).

### Character-sets

Trong Java, một character-set là một hệ thống mapping giữa Unicode character và bytes. Trước đây với Java IO, việc sử dụng character-set vẫn còn hạn chế và thiếu sót với character stream.

Java NIO cung cấp một bộ công cụ (encoder/decoder) để xác định character-set và cung cấp giải thuật encode/decode cho các lớp mapping.

### Regular Expressions

Việc tích hợp Regular Expression vào Java NIO là một bước cải tiến tuyệt vời. Trước đây việc tìm kiếm nội dung trong một file text phải trải qua nhiều bước và gặp phải rất nhiều khó khăn. Với việc cung cấp khả năng search bằng Regular Expression trực tiếp với Channel, moi việc trở nên dễ dạng hơn rất nhiều.

[CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI 1](#_Toc303935189)

[1.1) HIỆN TRẠNG 1](#_Toc303935190)

[1.1.1) Điện thoại di động ngày càng trở nên gắn bó với hoạt động hằng ngày của con người.’ 1](#_Toc303935191)

[1.1.2) Smartphone và thị trường lớn dành cho ứng dụng điện thoại di động 2](#_Toc303935192)

[1.1.3) Hệ điều hành Android – Sự bùng nổ của Smartphone. 5](#_Toc303935193)

[1.2) NHU CẦU 7](#_Toc303935194)

[1.2.1) Phát triển các ứng dụng thuần Việt dành cho Android 7](#_Toc303935195)

[1.2.2) Nhu cầu ứng dụng từ điển dành cho Android 8](#_Toc303935196)

[1.3) CÁC THÁCH THỨC 9](#_Toc303935197)

[1.3.1) Cần thời gian để tìm hiểu về kiến trúc và cách sử dụng API của Android 9](#_Toc303935198)

[1.3.2) Không có khả năng mua thiết bị Android thật sự 9](#_Toc303935199)

[1.3.3) Giới hạn khả năng vật lý của thiết bị 9](#_Toc303935200)

[1.3.4) Số lượng khổng lồ các ứng dụng Android trên Internet 10](#_Toc303935201)

[1.4) XÁC ĐỊNH ĐỀ TÀI 11](#_Toc303935202)

[1.4.1) Mục tiêu cơ bản 11](#_Toc303935203)

[1.4.2) Mục tiêu nâng cao 13](#_Toc303935204)

[1.4.3) Hướng phát triển 14](#_Toc303935205)

[CHƯƠNG 2 : CÁC KIẾN THỨC LIÊN QUAN 15](#_Toc303935206)

[2.1) Hệ điều hành di động (mobile operating systems) 15](#_Toc303935207)

[2.1.1) Giới thiệu 15](#_Toc303935208)

[2.1.2) Lịch sử 16](#_Toc303935209)

[2.1.3) Dự đoán về thị phần 17](#_Toc303935210)

[2.1.4) Các hệ điều hành phổ biến 18](#_Toc303935211)

[1) Android của Google Inc. (mã nguồn mở) 18](#_Toc303935212)

[2) BlackBerry của RIM (mã nguồn đóng) 19](#_Toc303935213)

[3) iOS của Apple Inc. (mã nguồn đóng) 20](#_Toc303935214)

[4) Symbian của Symbian Foundation (open public license) 21](#_Toc303935215)

[5) Windows Phone của Microsoft (mã nguồn đóng) 22](#_Toc303935216)

[6) webOS của HP 23](#_Toc303935217)

[2.2) Giới thiệu về hệ điều hành Android 24](#_Toc303935218)

[2.2.1) Lịch sử sáng lập và hoạt động 24](#_Toc303935219)

[2.2.2) Các phiên bản Android 25](#_Toc303935220)

[2.2.3) Kiến trúc Android 26](#_Toc303935221)

[2.2.4) Android Market 28](#_Toc303935222)

[2.2.5) Thị phần 29](#_Toc303935223)

[2.2.6) Thị phần theo phiên bản Android 29](#_Toc303935224)

[2.3) Môi trường lập trình Android 30](#_Toc303935225)

[2.3.1) Yêu cầu phần cứng 30](#_Toc303935226)

[2.3.2) Cài đặt Eclipse và khởi động emulator 30](#_Toc303935227)

[2.3.3) Tạo Android project và chạy thử 34](#_Toc303935228)

[2.4) Android emulator 36](#_Toc303935229)

[2.4.1) Khái quát 36](#_Toc303935230)

[2.4.2) Android Virtual Devices và Emulator 36](#_Toc303935231)

[2.4.3) Android emulator và những hạn chế 37](#_Toc303935232)

[2.5) Lập trình Android 38](#_Toc303935233)

[2.5.1) Đặc tính 38](#_Toc303935234)

[2.5.2) Kiến trúc 39](#_Toc303935235)

[1) Applications 39](#_Toc303935236)

[2) Application framework 40](#_Toc303935237)

[3) Libraries 41](#_Toc303935238)

[4) Android Runtime 41](#_Toc303935239)

[5) Linux kernel 42](#_Toc303935240)

[2.5.3) Kiến thức cơ bản 43](#_Toc303935241)

[1) Application components 44](#_Toc303935242)

[2) Activities 44](#_Toc303935243)

[3) Services 44](#_Toc303935244)

[4) Content providers 45](#_Toc303935245)

[5) Broadcast receivers: 45](#_Toc303935246)

[6) Activating components 46](#_Toc303935247)

[7) The manifest file 47](#_Toc303935248)

[8) Application resources 47](#_Toc303935249)

[2.5.4) Giới thiệu SQLite. 49](#_Toc303935250)

[2.5.5) SQLite trong MegaDict 49](#_Toc303935251)

[2.5.6) Thiết kế khả năng đáp lại của ứng dụng (Designing for responsiveness). 50](#_Toc303935252)

[2.5.7) Vòng đời của Activity: 54](#_Toc303935253)

[2.6) DICT Dictionary Format 56](#_Toc303935254)

[2.6.1) Giao thức DICT (DICT Protocol) 56](#_Toc303935255)

[2.6.2) Định dạng từ điển của DICT Protocol 57](#_Toc303935256)

[2.6.3) Sử dụng trực tiếp DICT Protocol database 60](#_Toc303935257)