ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

Tel. (84-511) 736 949, Fax. (84-511) 842 771

Website: [itf.ud.edu.vn](mailto:itf.ud.edu.vn), E-mail: cntt@edu.ud.vn

BÁO CÁO THỰC TẬP TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

MÃ NGÀNH: 05115

ĐỀ TÀI:

DU LỊCH VIỆT NAM

SINH VIÊN : Huỳnh Đức Dũng - 06T1

Nguyễn Văn Huỳnh – 06T2

ĐƠN VỊ : Công Ty Toàn Cầu Xanh

CBHD : Nguyễn Hữu Dũng

ĐÀ NẴNG, 01/2011

LỜI CẢM ƠN

Chúng tôi chân thành cảm ơn anh Nguyễn Hữu Dũng đã tận tình giúp đỡ, hướng dẫn, góp ý cho đề tài. Chúng tôi xin cảm ơn ban lãnh đạo và tập thể nhóm lập trình của công ty TNHH TT – KTS Toàn Cầu Xanh đã tạo điều kiện thuận lợi, hỗ trợ rất nhiều trong quá trình thực tập. Cảm ơn sự giúp đỡ quý báu của các bạn hữu về tài liệu và kinh nghiệm.

**LỜI CAM ĐOAN**

*Chúng tôi xin cam đoan:*

* 1. *Những nội dung trong báo cáo này là do chúng tôi thực hiện dưới sự hướng dẫn trực tiếp của anh Nguyễn Hữu Dũng.*
  2. *Mọi tham khảo dùng trong báo cáo này đều được trích dẫn rõ ràng tên tác giả, tên công trình, thời gian, địa điểm công bố.*
  3. *Mọi sao chép không hợp lệ, vi phạm quy chế đào tạo, hay gian trá, chúng  
     tôi xin chịu hoàn toàn trách nhiệm.*

*Sinh viên,*

*Huỳnh Đức Dũng*

*Nguyễn Văn Huỳnh*

**MỤC LỤC**

MỞ ĐẦU 3

I. Giới thiệu đơn vị thực tập 3

II. Giới thiệu đề tài 3

III. Mục đích đề tài 4

IV. Mục tiêu và kết quả đề tài 4

CAKEPHP 5

I. Giới thiệu 5

II. Cấu trúc thư mục của cake 5

III. Mô hình Model View Controller – MVC 6

IV. Cách làm việc của MVC 8

IV.1. Model 9

IV.2. Controller 9

IV.3. View 9

V. Quy ước đặt tên của cake 9

V.1. Cách đặt tên Controller 9

V.2. Cách đặt tên Model 10

V.3. Cách đặt tên View 10

V.4. Cách đặt tên bảng 10

VI. Components 10

VII. Helpers 10

VIII. Data Validation 12

IX. Quan hệ các bảng trong cakephp 12

J2ME 18

I. Giới thiệu 18

I.1. Khái quát lớp J2ME 18

I.2. Máy ảo Java (hay KVM) 19

I.3. Tầng CLDC (Connected Limited Device Configuration) 22

I.3.1. CLDC – Connected Limited Device Configuration 22

I.3.2. Sự khác nhau giữa J2ME và J2SE 23

I.4. Sự cần thiết của J2ME 25

I.5. Hiện trạng thiết bị thông tin di động (Mobile Information Device Profile) 25

I.6. Các kiểu ứng dụng MIDP 26

GOOGLE MAP API 28

GPS VÀ VẤN ĐỀ ĐỊNH VỊ TỌA ĐỘ 30

I. Hệ thống định vị toàn cầu – GPS là gì? 30

II. Lịch sử phát triển GPS 30

III. Phân loại 31

IV. Sự hoạt động của GPS và tín hiệu GPS 31

IV.1. Sự hoạt động của GPS 31

IV.2. Vệ tinh GPS xác định một điểm trên trái đất như thế nào? 33

IV.3. Các thành phần của GPS 37

IV.4. Tín hiệu GPS 38

IV.5. Các định dạng tín hiệu mà GPS Receiver nhận được 40

V. Độ chính xác của GPS và các nguồn lỗi của tín hiệu GPS 41

KẾT LUẬN 43

I. Kết quả đạt được 43

I.1. Ưu điểm 43

I.2. Nhược điểm 43

II. Hướng phát triển 43

CHƯƠNG 0

# MỞ ĐẦU

## Giới thiệu đơn vị thực tập

**Công ty TNHH Truyền Thông - Kỹ Thuật Số Toàn Cầu Xanh** được thành lập tại Việt Nam vào ngày 29/02/2008 cùng 2 chi nhánh tại Úc và Singapore với các ngành nghề hoạt động chủ yếu trong lĩnh vực CNTT như: thiết kế Web, phát triển các ứng dụng trên nền Web, tư vấn giải pháp phần mềm, thiết kế nhận diện thương hiệu. **Toàn Cầu Xanh** tự hào có được một đội ngũ chuyên viên, nhân viên giàu kinh nghiệm, kiến thức chuyên môn tốt, đa dạng. Đội ngũ tư vấn uy tín của công ty **Toàn Cầu Xanh** luôn đồng hành cùng khách hàng nhằm đưa ra những giải pháp tối ưu nhất, mang đến sự hài lòng cho khách hàng với những sản phẩm có chất lượng cao cùng với giá thành hợp lý.

Với mục tiêu trở thành một đơn vị có uy tín trong lĩnh vực CNTT tại Việt Nam và Úc, đem lại những tiện ích cho khách hàng, chúng tôi sẽ không ngừng nâng cao trình độ, củng cố đội ngũ, cải tiến nâng cao chất lượng sản phẩm để ngày càng có những sản phẩm tốt hơn phục vụ khách hàng.

## Giới thiệu đề tài

Việt Nam trong những năm gần đây có bước phát triển ấn tượng, tạo dựng được hình ảnh là điểm đến thân thiện, hấp dẫn du khách. Số lượng khách quốc tế đến Việt Nam đang ngày càng tăng. Nếu như năm 2000, Việt Nam đón 2,1 triệu lượt khách quốc tế, năm 2008 là 4,2 triệu lượt khách quốc tế và năm 2010, số khách du lịch đến Việt Nam sẽ vượt con số 5 triệu lượt, vượt xa mục tiêu ban đầu là 4,2 triệu lượt. Với tốc độ tăng trưởng trung bình hàng năm về lượng khách khoảng 20%, Việt Nam ngày càng khẳng định là một điểm đến hấp dẫn trên bản đồ du lịch thế giới. Số du khách nội địa năm 2010 cũng ước đạt 28 triệu lượt, thu nhập từ du lịch khoảng 96.000 tỷ đồng, đóng góp khoảng 5% GDP, giải quyết việc làm cho 1,4 triệu lao động.

Từ thực tiễn đó, nhóm chúng em mong muốn ứng dụng tin học vào lĩnh vực du lịch để góp phần du lịch Việt Nam ngày càng là một điểm đến yêu thích trên thế giới. Dự án “Du Lịch Việt Nam” là một hệ thống gồm website và phần mềm chạy trên di động. Khi du khách đến Việt Nam, với những chiếc smartphone trên tay thì khách du lịch dễ dàng định vị được tọa độ của mình nhờ qua GPS. Nếu du khách có cài phần mềm “Du Lịch Việt Nam” thì dễ dàng chia sẻ những hình ảnh mình chụp được lên website qua GPRS. Đồng thời, phần mềm trên di động còn cung cấp nhiều thông tin bổ ích về du lịch như: tìm đường, tìm máy ATM, các địa danh du lịch gần vị trí hiện tại, các quán ăn đặc sản. Với website, từ những hình ảnh cung cấp từ người dùng là du khách thì lại đem đến cho người dùng trải nghiệm du lịch qua ảnh rất thực tế. Website sẽ giống như một mạng xã hội về du lịch. Bạn đi du lịch, bạn chia sẻ hình ảnh lên website. Bạn bè của bạn vào xem, viết cảm nhận, chia sẻ link, bình chọn ảnh đẹp. Bên cạnh website còn có những chức năng tương tự như phần mềm trên di động (tìm đường, tìm ATM) thì còn liên kết đến với nhiều nhà hàng, khách sạn để cung cấp thông tin lên website, cho phép liên hệ đặt hàng online.

## Mục đích đề tài

Xây dựng hệ thống phần mềm “Du Lịch Việt Nam” trên website và điện thoại di động.

Quảng bá hình ảnh Việt Nam tới bạn bề trên toàn thế giới.

Mang lại lợi nhuận cho ngành kinh doanh du lịch Việt Nam

## Mục tiêu và kết quả đề tài

* Tìm hiểu Cakephp
* Tìm hiểu Java (J2ME)
* Tìm hiểu Google Map Api
* GPS

Qua dự án này, công việc cần đạt của nhóm là xây dựng để hệ thống tích hợp giữa website và điện thoại. Hệ thống cung cấp khả năng như sau:

* Dựa vào GPS, định vị được tọa độ của du khách.
* Tìm đường đi, tìm thông tin cần thiết (nhà hàng, khách sạn, bệnh viên, công viên).
* Xây dựng được webservices để ảnh chụp từ điện thoại được đăng lên website.
* Xây dựng được mạng xã hội du lịch trực tuyến

Theo chúng tôi, đây là một đề tài hay và thiết thực. Nhóm chúng tôi mong nhận được sự góp ý của các thầy cô hướng dẫn để thực hiện thành một đề tài tốt nghiệp.

CHƯƠNG 1

# CAKEPHP

## Giới thiệu

Cakephp là một framework phát triển nhanh của PHP dùng các mô hình thiết kế phổ biến được biết đến như Active Record, Association Data Mapping, Front Controller và MVC. Mục đích chính của chúng là cung cấp một cấu trúc framwork cho phép người dùng PHP ở mọi cấp độ phát triển một cách nhanh chóng một ứng dụng web mạnh mà không làm mất đi bất cứ tính linh hoạt.

Một framework PHP là một bộ ( collection ) các mã, thư viện, lớp và môi trường run-time giúp người phát triển xây dựng trang web nhanh hơn. Ý tưởng chính sau việc sử dụng các framworks là cung cấp cho người phát triển với các chức năng thông thường và một cấu trúc cơ bản để họ có thể xây dựng trên ứng dụng của họ. Hầu hết những người phát triển PHP, có vài kinh nghiệm và một tập các thư viện của họ và cấu trúc để giúp chúng phát triển nhanh hơn, vì họ có thể sử dụng các mã cơ bản của họ từ một project này tới các project khác. Nhưng, việc xây dựng một ứng dụng web trên một framework mã nguồn mở như CakePHP rõ ràng có những lợi ích của nó. Với nó, người phát triển không chỉ sử dụng kinh nghiệm của họ mà còn kinh nghiệm của nhiều người phát triển khác là những người đã dùng và phát triển framework. Framework cũng tin cậy hơn vì nó được dùng và kiểm tra bởi nhiều người phát triển ứng dụng. Bên cạnh đó, dùng một framework như CakePHP có nhiều thuận lợi mà tất cả người phát triển liên quan trong project phải theo qui ước và qui tắc giống nhau để cấu trúc ứng dụng. Điều này trở nên dễ dàng hơn cho những người phát triển mới để giải quyết nếu chúng cùng qui ước của framework.

## Cấu trúc thư mục của cake

Khi bạn mở gói Cake, bạn sẽ thấy được các folders chính sau:

* app: Chứa các file và folders cho ứng dụng của bạn. Thư mục app là thư mục phát triển ứng dụng của bạn, chứa các foder và các files bên trong.
* cake: Chứa các thư viện core Cake. Bạn không nên đụng vào những thư mục này trừ khi bạn biết bạn đang làm gì.
* docs: Chứa các file tài liệu Cake chẳng hạn như: read me, copy right, và thay đổi các file log text. Bạn có thể lưu trữ tài liệu của chính bạn trong thư mục này.
* venders: Chứa party-code. Thư mục này có thể chứa các thư viện third-party, chẳng hạn như gói Swift Mailer cho việc gởi một tin email.

Việc chia thư mục thư viện mặc định core Cake từ thư mục ứng dùng làm cho có khả năng để nhiều ứng dụng khác nhau chia sẻ một bản cài đặt Cake. Với cấu trúc thư mục này, bạn có thể dễ dàng nâng cấp phiên bản Cake đã tồn tại mà không ảnh hưởng tới bất cứ ứng dụng nào bạn đã viết. Bảng sau mô tả chi tiết cấu trúc thư mục Cake mặc định.

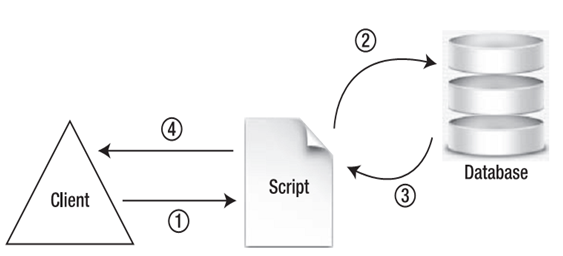
|  |  |
| --- | --- |
| Thư mục | Mô tả |
| app    config/    controllers/    components/  /index.php  locale/  models/  plugins/  tests/    tmp/  vendors/  views/    elements/  errors/  helpers/  layouts/  webroot/  css/  files/  img/  js/ | * Thư mục cha cho ứng dụng của bạn. * Chứa các file cấu hình cho cấu trúc chung chẳng hạn như: kết nối cơ sở dữ liệu, bảo mật, truy cập control. * Chứa các file controllers trong ứng dụng( ví dụ : users\_controller.php). * Chứa các files user-defined component. * Cho phép bạn deploy-triển khai Cake với /app như DocumentRoot. * Chứa các locale(nơi diễn ra) file- xử lý với quốc tế hóa. * Chứa các file model. * Chứa các files plugin. * Chứa các thư mục và các files test. * Dùng cho caches và logs. * Chứa các thư viện thirth-party. * Chứa các thư mục và các files view cho việc hiển thị (e.g: show.ctp). * Elements. * Custom error pages. * Helpers. * Các file layout ứng dụng. * DocumentRoot cho ứng dụng. * Chứa các file ứng dụng style sheet. * Chứa bất cứ files nào. * Chứa đồ họa. * Chứa các file JavaScript. |
| cake/ | * Chứa các thư viện core Cake. |
| vendors/ | * Chứa các thư viện thirt-party cho tất cả các ứng dụng. |

*Figure 1: Cấu trúc thư mục Cake*

## Mô hình Model View Controller – MVC

Cake tuân theo cấu trúc MVC cho ứng dụng Web của bạn. Đây là mô hình thiết kế được dùng thông thường trong phát triển phần mềm, ở đó code được chia làm 3 phần chính: models, views, controllers. Models cho toàn bộ tương tác với database, views cho việc xuất ra và hiển thị, controllers cho tất cả các lệnh hay scripts cho nhập vào và program flow. Một ứng dụng điển hình PHP trộn những chức năng này trong cùng một code, làm cho nó khó duy trì (maintain) và debug.

Đây là dòng truyền ( flow) điển hình cho PHP scripting:



*Figure 2: The typical flow for PHP scripting*

1. Client gởi yêu cầu đến một PHP script bằng việc gõ một URL hoặc click vào một link.

2. Script xử lý dữ liệu và sau đó gởi yêu cầu dữ liệu trực tiếp tới database.

3. Scritpt nhận bất cứ dữ liệu ra và xử lý dữ liệu.

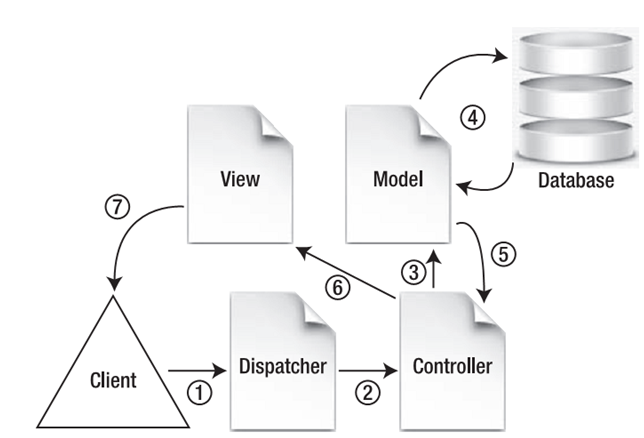
4. Script tạo ra output và forward nó tới trình duyệt của client.

Nói tóm lại, mọi thứ được chứa trong một PHP script. Bằng việc sử dụng hàm include(), người phát triển cởi bỏ tất cả các hàm thông thường vào các file ngoài khác, làm cho nó có thể giảm được sự dư thừa. Các ứng dụng PHP phức tạp nhất dùng các đối tượng có thể gọi bất cứ đâu trong ứng dụng đó, và chỉnh sửa tùy thuộc vào các biến và các thiết lập chuyển vào chúng. Người phát triển khi dùng các đối tượng và các lớp có thể cấu trúc ứng dụng theo nhiều cách.

MVC phát triển dựa trên PHP flow, và là một kĩ thuật hiệu quả trong việc tạo ra các đối tượng lớp hiệu dùng trong toàn bộ ứng dụng. Mục tiêu chính đằng sau MVC là tạo ra mỗi chức năng của ứng dụng được viết một lần và chỉ một lần, vì thế, dòng code được giảm dư thừa. Cake đạt được mục đích này bằng việc không chỉ cung cấp tài nguyên cho MVC có thể làm được, mà còn bằng việc sử dụng một phương pháp nhất quán cho nơi để lưu trữ các hoạt động trong ứng dụng. Đơn giản đặt tên các file của bạn một cách chắc chắn cho phép Cake hợp nhiều các tài nguyên với nhau mà không dùng bất cứ mã chỉ định nào.

## Cách làm việc của MVC

MVC có thể khác nhau tùy thuộc vào framework mà bạn đang làm việc, nhưng nói chung nó làm việc như sau:



*Figure 3: How Cake makes use of the MVC structure*

1. Client gởi yêu cầu một trang tới ứng dụng, bằng việc gõ một URL hay click vào một link. Theo qui ước, một URL điển hình thường có cấu trúc:

http://{Domain}.com/{Application}/{Controller}/{Action}/{Parameter 1, etc.}

2. Dispatcher (người gởi đi) script phân tích cấu trúc URL và quyết định controller nào để thực thi. Nó cũng có thể chuyển (pass) cùng với bất cứ hành động và các tham số tới controller.

3. Hàm trong controller có thể cần để xử lý nhiều dữ liệu hơn các tham số forwarded bởi dispatcher. Nó có thể chạy nhiều truy vấn với cơ sở dữ liệu và sắp xếp dữ liệu.

5. Khi model lấy (pull) bất cứ dữ liệu từ hoặc gởi dữ liệu đến database, nó trả về output của nó tới controller

6. Controller xử lý dữ liệu và xuất tới file view.

7. View thêm vào bất cứ thiết kế hoặc hiển thị dữ liệu tới controller, xuất và gởi việc xuất của nó tới trình duyệt của client.

Lợi ích của việc sử dụng MVC để phát triển các trang web là các chức năng được lặp lại và các nhiệm vụ có thể được chia cắt, vì vậy cho phép chỉnh sửa nhanh hơn. Thậm chí nó có thể giúp trong việc gỡ rối. Một lỗi được giữ trong suốt quá trình tương tác với database. Thường thường, vấn đề sẽ ở nơi nào đó trong một model. Hiểu được tương tác database xảy ra chỉ một nơi làm cho nó dễ dàng giải quyết những vấn đề.

### Model

Trong CakePHP, một model hiển thị một bảng cơ sở dữ liệu cụ thể. Liệu có một model. Tất cả mã PHP liên quan đến truy cập, thêm, chỉnh sửa hay xóa các bản ghi từ bảng đều ở trong (situated ) model. Models cũng chứa các code định nghĩa quan hệ của nó với các models khác. Nó cũng định nghĩa các qui tắc điều kiện dữ liệu khi thêm và cập nhật dữ liệu cho model đó. Model có thể được hiểu như là tầng dữ liệu của ứng dụng. Model cũng là nơi xảy ra business logic liên quan tới model đó nên được định nghĩa. Ví dụ, nếu ta có một model để hiển thị các xe hơi, tất cả các hành động liên quan đến nó như mua xe, bán xe, nên được định nghĩa trong model đó.

### Controller

Trong CakePHP, controllers điều khiển dòng ứng dụng hay logic của ứng dụng. Mỗi trang web yêu cầu được trực tiếp đến một controller cụ thể nơi mà người dùng nhập vào (dữ liệu POST hoặc GET ) được chấp nhận. Sau đó controller logic chấp nhận đáp ứng gì được tạo ra. Controller logic thường chứa các lời gọi tới các models để truy cập dữ liệu, và cũng có các chức năng như kiểm tra, truy cập, điều khiển. Cuối cùng, controller chuyển đáp ứng(output ) tới view. Controller có thể được hiểu như tầng logic của ứng dụng. Như đã đề cập ở trên, model nên có tất cả business logic của một ứng dụng. Controllers nên ủy nhiệm các hoạt động tới model. Thiết kế khoa học thường được nói đến như “fat models and thin controllers”.

### View

Views là các đầu ra(output) hoặc các đáp ứng được gởi trở lại tới người dùng khi một yêu cầu được xử lý. Cơ bản, chúng gồm mã markup (như HTML) nhúng với mã PHP, nhưng chúng cũng có thể có các form khác cho đầu ra như XML, PD tùy thuộc vào hoàn cảnh. Views có thể được hiểu như là tầng hiển thị của ứng dụng.

## Quy ước đặt tên của cake

### Cách đặt tên Controller

Tên lớp phải là số nhiều và phải có Controller được nối theo sau, ví dụ UsersController. Nếu đối tượng có nhiều hơn một từ thì từ thứ hai phải bắt đầu với một chữ hoa, như là OnlineUsersController. Không dùng kí tự gạch dưới để chia cắt các từ.

Tên file phải là số nhiều với \_controller nối theo và đuôi file là .php, ví dụ như users\_controller.php. Nếu đối tượng có nhiều hơn một từ thì theo sau các từ là dấu gạch dưới, ví dụ như online\_users\_controller.php.

### Cách đặt tên Model

Tên lớp phải là số ít, như User. Nếu đối tượng có nhiều hơn một từ thì từ thứ hai phải bắt đầu bằng kí tự hoa, như OnlineUser.

Tên file phải là số ít, với đuôi .php, như user.php. Nếu đối tượng có nhiều hơn một từ thì theo sau các từ là dấu gạch dưới, như online\_user.php.

### Cách đặt tên View

Tên file views được lấy bởi tên của hành động( action) trong controller. Ví dụ như đối tượng có một phương thức là UsersController::show(), thì đường dẫn là app/views/users/show.ctp.

### Cách đặt tên bảng

Tên bảng trong cơ sở dữ liệu phải là số nhiều, với các từ được chia ra bởi dấu gạch dưới, như country\_codes. Bạn có thể ghi đè qui ước đặt tên này bằng việc thiết lập thuộc tính $useTable cho tên bảng mà bạn muốn.

Ví dụ, bạn có thể thiết lập như sau:

$useTable=’mytable’;

Với mytable là tên của một bảng trong cơ sở dữ liệu.

## Components

Components là các lớp được định nghĩa để thực thi các nhiệm vụ ứng dụng cụ thể để hỗ trợ controller. Cake xây dựng sẵn nhiều components, chẳng hạn như Auth (cho xác thực người dùng và nhóm), Email, Session, RequestHandler, Acl( cho điều khiển truy cập người dùng và nhóm). Components có được được định nghĩa bởi người dùng.

Tất cả các components mà bạn phát triển phải được lưu trữ trong thư mục app/controllers/components.

## Helpers

Helpers là các lớp hỗ trợ để giảm thời gian phát triển bằng việc cung cấp các shortcuts để tạo ra các thành phần hiển thị. Các file helper được lưu trữ trong thư mục app/views/helpers. Bảng sau mô tóm tắt vài helpers có sẵn trong Cake. Để tìm hiểu kĩ hơn về Cake helpers, hãy vào link Cake API tại trang <http://api.cakephp.org>.

|  |  |
| --- | --- |
| Helper | Mô tả |
| HTML | Giúp tạo ra tự động các thành phần HTML và cũng cho phép tạo ra các thẻ HTML bằng việc chấp nhận và phân tích các biến. Helper này được gọi trong view bằng việc sử dụng đối tượng $html. Để bao gồm một tham chiếu tới helper HTML, dùng biến $helpers=array(‘Html’). |
| Form | Giúp tạo ra các form và xử lý. Dùng đối tượng $form với các hàm của nó để tạo các thành phần form. |
| Ajax | Giúp để đơn giản các nhiệm vụ Ajax. Nó yêu cầu câu lệnh $javascript->link(array(‘prototype’))trong view, tham chiếu Prototype JavaScript framework, để làm việc đúng cách trong một view. |
| JavaScript | Giúp đơn giản các nhiệm vụ JavaScript, như là tạo ra đối tượng JavaScript Object Notation (JSON) từ một mảng, dùng $javascript->object(). Để gắn một sự kiện với một thành phần dùng $javascript->event(). |
| Paginator | Giúp định dạng dữ liệu thành nhiều trang hay sắp xếp dữ liệu dựa trên vài tham số. |
| Session | Cung cấp các hàm để xử lý với trình quản lý session. Ví dụ, để đưa ra một message,dùng $session->flash(). Để đọc tất cả các giá trị lưu trữ trong một session cho trước, dùng  $session->read(). |
| Text | Cung cấp các hàm để xử lú với text hoặc string. |
| Time | Giúp quản lý date và time. |
| XML | Giúp hiệu chỉnh XML. Để tạo ra các thành phần XML, dùng hàm $xml->elem(). |

*Figure 4: Helpers*

## Data Validation

Data Validation là một phần thiết yếu của đảm bảo toàn vẹn và chính xác dữ liệu submitted bởi người dùng thông qua một web form. Cake xây dựng vài kĩ thuật tiêu chuẩn. Bạn chỉ định các qui tắc trong một model và Cake sẽ tự động áp dụng các qui tắc đó khi một web form được kết nối với model đó. Các qui tắc đó cũng có thể áp dụng cho dữ liệu XML.

## Quan hệ các bảng trong cakephp

Trong CakePHP, quan hệ giữa các bảng trong cơ sở dữ liệu được định nghĩa thông qua sự kết hợp(association). Bằng việc sử dụng CakePHP ORM (object relational mapping), ta có thể lưu, truy lục, xóa dữ liệu liên quan vào hoặc từ nhiều bảng khác nhau đơn giản và nhanh hơn mà không cần phải viết nhiều câu lệnh truy vấn SQL phức tạp với nhiều câu lệnh JOIN nữa.

Ta đã biết có 3 kiểu quan hệ cơ bản giữa các bảng trong cơ sở dữ liệu:

1. one-to-one (has one)

2. one-to-many (has many)

3. many-to-many (has and belongs to many)

Để tìm hiểu về các mối quan hệ này chúng ta xây dựng một ứng dụng blog đơn giản dùng scaffolding để kiểm tra. Ứng dụng này sẽ giúp chúng ta thảo luận về các kết hợp trong một cách dễ dàng hơn dùng Cake. Tạo một ứng dụng Cake tên blog.

Thiết kế dữ liệu:

CREATE TABLE `posts` (

`id` int(11) unsigned NOT NULL auto\_increment,

`name` varchar(255) default NULL,

`date` datetime default NULL,

`content` text,

`user\_id` int(11) default NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `comments` (

`id` int(11) unsigned NOT NULL auto\_increment,

`name` varchar(100) default NULL,

`content` text,

`post\_id` int(11) default NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `users` (

`id` int(11) unsigned NOT NULL auto\_increment,

`name` varchar(100) default NULL,

`email` varchar(150) default NULL,

`firstname` varchar(60) default NULL,

`lastname` varchar(60) default NULL,

PRIMARY KEY (`id`)

);

Bảng users sẽ chứa danh sách các tác giả hiện có các bài blog. Khi một bài viết được tạo ra, nó sẽ được gán một tác giả từ bảng users. Chú ý rằng, có một trường trong bảng posts tên user\_id. Cái này sẽ đúng(match) với trường id trong bảng users, nó liên kết một tác giả với bài viết. Tương tự, trong bảng comment, mỗi chú thích sẽ được gán với một bài viết trong cùng một kiểu tương tự. Trong trường hợp này có một trường tên post\_id, sẽ đúng (match) với trường id trong bảng posts.

Trong các models, bạn sẽ giải thích rõ ràng(spell out) các kết hợp này để Cake có thể kéo(pull) chúng với nhau. Hơn nữa, bạn có thể kiểm tra tốt cách bạn vừa chỉ định các kết hợp này dùng đặc tính scaffolding.

**“Belongs To”**

Khi kết hợp các bảng, bạn cần nói cho Cake kiểu quan hệ gì mỗi bảng với các bảng khác. Blog sẽ có một quan hệ “belongs to” trong một bộ đôi (a couple ) các bảng. Đầu tiên, mỗi bài viết blog sẽ được gán với một tác giả đã chỉ đinh, mỗi bài post blog “belongs to” một user. Bạn đặt một trường user\_id trong bảng posts để lưu mối quan hệ này. Mỗi bảng ghi trong bảng posts, một trong các bảng ghi từ bảng user sẽ được lưu lại bằng việc gán một IDs của nó tới user\_id.

Để xây dựng quan hệ này trong models, đầu tiên trong model Post (app/models/post.php) ta thêm vào các dòng code để gán nó một quan hệ “belongs to” với User model:

<?

class Post extends AppModel {

var $name = 'Post';

var $belongsTo = array('User');

}

?>

Bạn làm điều này bằng cách gán một mảng của các model đó là phần của quan hệ tới model hiện tại. Biến đối tượng *class* Cake dùng để xây dựng quan hệ “belongs to” là thuộc tính var $belongsTo.

Trong bất cứ ứng dụng Cake, mối quan hệ “belongs to” được tạo ra bởi những dòng code trên. Bạn cũng có thể thêm vào các mối quan hệ bằng cách include chúng trong một cú pháp (syntax) mảng.

**className**

Một từ khóa có thể được bao gồm với thiết lập $belongsTo là className. Đơn giản, className là model mà hiện tại model thuộc về. Trong trường hợp này, nó sẽ được thiết lập là User, nghĩa là tên lớp của model của bảng users.

**foreignKey Parameter**

Từ khóa này thiết lập khóa ngoại tìm thấy trong model liên quan. Việc thiết lập này hữu ích cho việc chỉ định nhiều mối quan hệ “belongs to”.

**Conditions Parameter**

Từ khóa này chứa một chuỗi SQL lọc các bảng ghi model liên quan. Nói chung, nó sẽ chứa câu lệnh SQL equals/not-equals cho một trường(ví dụ, Post.published=1).

**Fields Parameter**

Mặc định, Cake sẽ trả về tất cả các trường từ bảng kết hợp. Trong trường hợp này, các trường từ bảng ghi User, được kết hợp với post hiện tại, sẽ được đưa ra và làm cho hiệu dụng(available) với model Post. Bạn có thể hạn chế điều này bằng việc dùng từ khóa fields.

Bạn có thể thiết lập những từ khóa này cho các giá trị của bạn bằng việc gán chúng như là một mảng với mỗi đối tượng trong mảng $belongsTo. Đoạn code sau sẽ chỉ ra quan hệ Post “belongs to” với tất cả các từ khóa được hiển thị:

var $belongsTo = array(

'User'=>array(

'className'=>'User',

'foreignKey'=>'user\_id',

'conditions'=>null,

'fields'=>null

)

);

**“Has One”**

Mỗi quan hệ với kết hợp ánh xạ phải được chỉ định trong cả hai hướng. Mỗi bài post thuộc về các user là chưa đủ. Bạn phải chỉ định trong model User cách các users được kết hợp với bất cứ bảng khác. Trong trường hợp này, một user sẽ có nhiều posts. Ba mối quan hệ là “has one”,”has many”,”has and belongs to many”.

Mối quan hệ “has one” này thực chất là quan hệ one-to-one. Trong nhiều ứng dụng gán profiles tới users, chẳng hạn các trang web, mối quan hệ “has one” được dùng. Mỗi user có một profile(hiện trạng), và một profile thuộc về chỉ một user.

Để thiết lập mối quan hệ “has one”, bạn thiết lập thuộc tính $hasOne như cách bạn đã làm với $blongsTo trong model Post. Thiết lập quan hệ :

var $hasOne = array('Post');

**className Parameter**

Trong mối quan hệ “has one”, className luôn luôn được thiết lập với model mà chứa thuộc tính belongsTo chỉ vào(point) vào model hiện tại.

**foreignKey,conditions,and fields Parameters**

Giống như trong “belongs to”

**dependent Parameter**

Khi xóa các bảng ghi trong quan hệ “has one”, có thể bạn muốn cả hai chiều kết hợp được xóa. Ví dụ, khi một user có một profile và user đó bị xóa, bạn muốn profile kết hợp cũng bị xóa theo. Trong trường hợp này, từ khóa dependent sẽ cho phép bạn làm điều này một cách dễ dàng. Mặc định, nó được thiết lập false. Thiết lập dependent thành true để xóa tất cả các trường trong cả 2 bảng khi hành động xóa được chạy thông qua các model kết hợp.

**“Has Many”**

Bạn vừa tạo ra Post model, giờ tạo ra User model. Trong thư mục app/models, tạo ra user model với dòng code:

<?

class User extends AppModel {

var $name = 'User';

var $hasMany = array('Post');

}

?>

Trong blog, mỗi user sẽ có nhiều bài posts. Thậm chí nếu user chỉ nhập vào một bài post, bạn vẫn muốn quan hệ này sẽ có thể lưu trữ nhiều hơn một post trên user.

Bằng việc nói cho User model rằng nhiều post records được kết hợp với nó, và hoàn chỉnh quan hệ trong model Post với thuộc tính belongsTo, Cake có thể liên kết 2 model với nhau.

Để xử lý nhiều hơn, bạn muốn thêm vào các tham số cho mối quan hệ này:

**className,foreignKey,conditions,and fields Parameters**

Các tham số này được mô tả ở trên.

**dependent Parameter**

Trong quan hệ “has many”, thiết lập dependent thành true sẽ làm việc một cách đệ quy. Nói cách khác, nếu bạn thiết lập thuộc tính $hasMany trong User model thành dependent=>true, thì bất cứ khi nào một user bị xóa, tất cả các post được gán với user đó sẽ bị xóa.

**Order Paramter**

Bạn có thể xử lý việc sắp xếp thứ tự các bảng ghi kết hợp bằng việc thêm vào cú pháp SQL trong tham số này. Ví dụ, trong User model, bạn có thể thiết lập order thành Post.datetime ASC, và nó sẽ sắp xếp tất cả các bài viết liên kết posts với ngày giờ theo thứ tự tăng dần.

**limit Parameter**

Vài cơ sở dữ liệu yêu cầu trả về một số lượng các bảng ghi kết hợp(asociated). Bạn có thể hạn chế số lượng các bảng ghi trả về để cắt giảm thời gian load trên server. Bạn làm điều này bằng việc thiết lập thuộc tính này với một giá trị hiển thị số lượng tối đa các bảng ghi kết hợp Cake sẽ đưa ra từ cơ sở dữ liệu.

Mối quan hệ “has many ” cực kỳ hữu ích cho việc giúp đỡ để thiết kế hiệu quả database. Nếu bạn biết rằng bạn dự định đặt một menu chọn trong ứng dụng của bạn cho nhiều lựa chọn sẽ được lưu trữ trong database, mối quan hệ này sẽ giúp bạn làm điều đó mà không phải viết một danh sách tĩnh trong HTML. Thay vào đó, bạn có thể xây dựng một bảng để lưu trữ những lựa chọn này và kết hợp chúng thông qua các models dùng quan hệ “has many”.

**Kiểm tra các kết hợp này:**

Một cách đơn giản để kiểm tra đó là dùng thuộc tính scaffolding. Bạn đã tạo ra Post và User model, giờ bạn cần tạo ra các controller để chạy scaffold.

Tạo file posts\_controller.php trong thư mục app/controllers, và thêm vài dòng code:

<?

class PostsController extends AppController {

var $name = 'Posts';

var $scaffold;

}

?>

Tạo ra file app/controllers/users\_controller.php và thêm vào vài dòng code:

<?

class UsersController extends AppController {

var $name = 'Users';

var $scaffold;

}

?>

**“Has and Belongs to Many”**

Đây là mối quan hệ có sức mạnh lớn (powerful) nhưng cũng có chút khó khăn để kiểm soát (master).Nhiều trang web dùng tags (thẻ) để thu xếp (order) nội dung của họ. Nhiều blog, gán một category với một post, sẽ có một danh sách các tags có thể được gán nhiều lần với nhiều câu chuyện, và ngược lại. Đây là mối quan hệ nhiều nhiều giữa posts và tags. Một post có thể có nhiều tags, và mỗi tag có thể thuộc về nhiều posts.

Cơ sở dữ liệu có thể xử lý các mối quan hệ many-to-many nếu có một bảng thứ ba lưu trữ mối quan hệ này. Trong ví dụ post-to-tag, một bảng thứ 3 tên posts-tags sẽ được tạo ra. Nó sẽ chỉ có hai trường là post\_id và tag\_id.

Để thêm mối quan hệ “has and belongs to may” vào ứng dụng blog, ta cần tạo ra một bảng mới tên tags trong cơ sở dữ liệu:

CREATE TABLE `tags` (

`id` int(11) unsigned NOT NULL auto\_increment,

`name` varchar(100) default NULL,

`longname` varchar(255) default NULL,

PRIMARY KEY (`id`)

);

Bảng này sẽ lưu trữ thông tin (category ) loại tags để tổ chức tốt hơn. Trường name sẽ là trường alphanumeric được sử dụng trong việc truy cập tag thông qua URL. Trường longname sẽ lưu trữ các title của category cho việc dùng links và page headings.

Vì bảng tags sẽ liên kết với bảng posts trong mối quan hệ “has and belongs to many”, bạn phải tạo ra một bảng khác để lưu giữ các kết hợp này. Tên bảng này sẽ theo quy tắc đặt tên của Cake. Đó là tên phải được sắp xếp theo thứ tụ với mỗi tên các bảng kết hợp được chia cách bởi kí tự gạch dưới. Bên trong bảng này, bạn cung cấp khóa ngoại và khóa ngoại kết hợp như các trường với các tên theo quy ước đặt tên cho mối quan hệ one-to-many. Trong trường hợp này, các trường tên được đặt theo thứ tự alphabetically với post\_id đầu tiên và tag\_id thứ hai. Bảng mới sẽ là posts\_tags:

CREATE TABLE `posts\_tags` (

`id` int(11) unsigned NOT NULL auto\_increment,

`post\_id` int(11) unsigned default NULL,

`tag\_id` int(11) unsigned default NULL,

PRIMARY KEY (`id`)

);

Tạo ra app/models/tags.php:

<?

class Tag extends AppModel {

var $name = 'Tag';

var $hasAndBelongsToMany = array('Post');

}

?>

Chỉnh sửa trong model Post:

<?

class Post extends AppModel {

var $name = 'Post';

var $belongsTo = array('User');

var $hasAndBelongsToMany = array('Tag');

}

?>

Cuối cùng, tạo ra Tags controller với scaffolding để bạn có thể add tags để kiểm tra quan hệ “has and belongs to many”.

<?

class TagsController extends AppController {

var $name = 'Tags';

var $scaffold;

}

?>

Để xử lý tốt hơn trên quan hệ này, dùng các tham số sau:

**className Parameter**

Tên này tương ứng với model kết hợp. Trong ví dụ Post::Tags, model Post sẽ cần className được thiết lập thành Tag.

**joinTable Parameter**

Cần nhớ rằng, cách bạn tạo ra bảng thứ 3 để chứa tất cả các kết hợp cho posts và tags. Điều này được ám chỉ đến là join table và có thể được thiết lập dùng tham số joinTable. Bạn sẽ thiết lập posts\_tags.

**foreignKey and associationForeignKey Parameters**

Khóa của model hiện tại trong bảng nối là associationForeignKey, trong trường hợp này sẽ là post\_id cho Post model. Tham số foreignkey là cho các model khác, đó là tag\_id cho Post model.

conditions,fields,order,and limit Parameters

Những tham số này được xử lý giống như các kết hợp khác.

CHƯƠNG 2

# J2ME

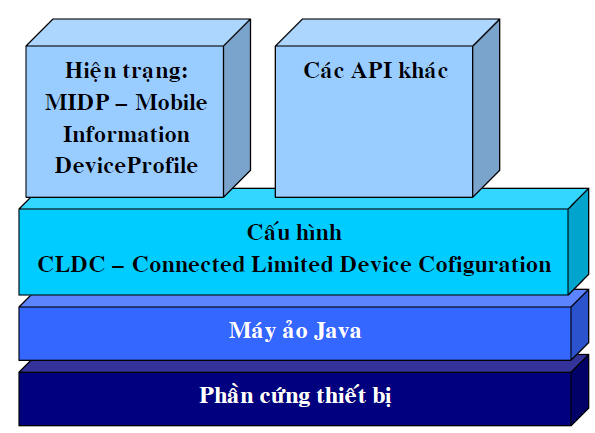
## Giới thiệu

J2ME (Java 2 Platform Micro Edition) là nền tảng Java, phiên bản thu nhỏ của Sun Microsystems, J2ME được xây dựng nhằm mang đến khả năng phát triển ứng dụng đa dạng, phong phú cho các thiết bị di động. Với ưu thế của ngôn ngữ Java, dựa trên hạ tầng mạng có sẵn của WAP, J2ME có thể dùng để xây dựng các ứng dụng từ đơn giản đến phức tạp nếu kết hợp với các công nghệ phía server.

J2ME hỗ trợ các thiết bị “micro” đa dạng, mà nó gọi là các “hiện trạng” (profile) nhưng tất cả chúng đều kém khả năng hơn so với máy tính cá nhân. Trong J2ME, sức mạnh CPU, bộ nhớ, lưu trữ và khả năng kết nối đều bị hạn chế, có thể là rất nghiêm ngặt.

### Khái quát lớp J2ME

Mục tiêu của J2ME là cho phép người lập trình viết các ứng dụng độc lập với thiết bị di động, không cần quan tâm đến phần cứng thật sự. Để đạt được mục tiêu này, J2ME được xây dựng bằng các tầng (layer) khác nhau để giấu đi việc thực hiện phần cứng khỏi nhà phát triển. Sau đây là các tầng của J2ME được xây dựng trên CLDC:



*Figure 5: Các tầng của CLDC J2ME*

Mỗi tầng ở trên tầng hardware là tầng trừu tượng hơn cung cấp cho lập trình viên nhiều giao diện lập trình ứng dụng (API – Application Program Interface) thân thiện hơn.

Tầng dưới lên trên:

Tầng phần cứng thiết bị (Device Hardware Layer)

Đây chính là thiết bị di động thật sự với cấu hình phần cứng của nó về bộ nhớ và tốc độ xử lý. Dĩ nhiên thật ra nó không phải là một phần của J2ME nhưng nó là nơi xuất phát. Các thiết bị di động khác nhau có thể có các bộ vi xử lý khác nhau với tập mã lệnh khác nhau. Mục tiêu của J2ME là cung cấp một chuẩn cho tất cả các loại thiết bị di động khác nhau.

Tầng máy ảo Java (Java Virtual Machine Layer)

Khi mã nguồn Java được biên dịch, nó được chuyển đổi thành mã bytecode. Mã bytecode này sau đó được chuyển thành mã ngôn ngữ máy của thiết bị di động. Tầng máy ảo Java bao gồm KVM (K Virtual Machine) là bộ biên dịch mã bytecode có nhiệm vụ chuyển mã bytecode của chương trình Java thành ngôn ngữ máy để chạy trên thiết bị di động. Tầng này cung cấp một sự chuẩn hóa cho các thiết bị di động để ứng dụng J2ME sau khi đã biên dịch có thể hoạt động trên bất kỳ thiết bị di động nào có J2ME KVM.

Tầng cấu hình (Configuration Layer)

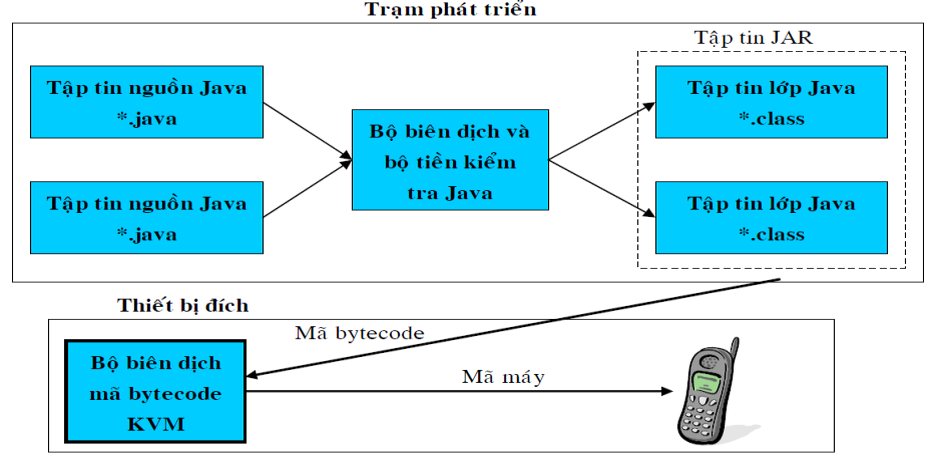
Tầng cấu hình của CLDC định nghĩa giao diện ngôn ngữ Java (Java Language Interface) cơ bản để cho phép chương trình Java chạy trên thiết bị di động. Đây là một tập các API định nghĩa lõi của ngôn ngữ J2ME. Lập trình viên có thể sử dụng các lớp và phương thức của các API này, tuy nghiên tập các API hữu dụng hơn được chứa trong tầng hiện trạng (Profile Layer).

Tầng hiện trạng (Profile Layer)

Tầng hiện trạng hay MIDP (hiện trạng thiết bị thông tin di động – Mobile Information Device Profile) cung cấp tập các API hữu dụng hơn cho lập trình viên. Mục đích của hiện trạng là xây dựng trên lớp cấu hình và cung cấp nhiều thư viện ứng dụng hơn. MIDP định nghĩa các API riêng biệt cho thiết bị di động. Cũng có thể có các hiện trạng và các API khác ngoài MIDP được dùng cho ứng dụng. Ví dụ, có thể có hiện trạng PDA định nghĩa các lớp và phương thức hữu dụng cho việc tạo các ứng dụng PDA (lịch, sổ hẹn, sổ địa chỉ). Cũng có thể có một hiện trạng định nghĩa các API cho việc tạo các ứng dụng Bluetooth. Thực tế, các hiện trạng kể trên và tập các API đang được xây dựng. Chuẩn hiện trạng PDA là đặc tả JSR – 75 và chuẩn Bluetooth API là đặc tả JSR – 82 với JSR là viết tắt của Java Specification Request.

### Máy ảo Java (hay KVM)

Vai trò của máy ảo Java hay KVM là dịch mã bytecode được sinh ra từ chương trình Java đã biên dịch sang ngôn ngữ máy. Chính KVM sẽ chuẩn hóa output của các chương trình Java cho các thiết bị di động khác nhau có thể có bộ vi xử lý và tập lệnh khác nhau. Không có KVM, các chương trình Java phải được biên dịch thành tập lệnh cho mỗi thiết bị di động. Như vậy lập trình viên phải xây dựng nhiều đích cho mỗi loại thiết bị di động. Hình sau đây biểu diễn tiến trình xây dựng ứng dụng MIDlet hoàn chỉnh và vai trò của KVM.



*Figure 6: Tiến trình xây dựng MIDlet*

Quá trình phát triển ứng dụng MIDlet với IDE (Môi trường phát triển tích hợp – Integrated Development Enviroment):

Lập trình viên: Tạo các tập tin nguồn Java

Bước đầu tiên là lập trình viên phải tạo mã nguồn Java, có thể có nhiều tập tin (\*.java).

Trên IDE: Bộ biên divhj Java (Java Complier): Biên dịch mã nguồn thành mã bytecode

Bộ biên dịch Java sẽ biên dịch mã nguồn thành mã bytecode. Mã bytecode này sẽ được KVM dịch thành mã máy. Mã bytecode đã biên dịch sẽ được lưu trong các tập tin \*.class và sẽ có một tập tin \*.class sinh ra cho mỗi lớp Java.

Trên IDE: Bộ tiền kiểm tra (Preverifier): Kiểm tra tính hợp lệ của mã bytecode

Một trong những yêu cầu an toàn của J2ME là bảo đảm mã bytecode chuyển cho KVM là hợp lệ và không truy xuất các lớp hay bộ nhớ ngoài giới hạn của chúng. Do đó tất cả các lớp đều phải được tiền kiểm tra trước khi chúng có thể được download về thiết bị di động. Việc tiền kiểm tra được xem là một phần của môi trường phát triển làm cho KVM có thể được thu nhỏ hơn. Bộ tiền kiểm tra sẽ gán nhãn lớp bằng một thuộc tính (attribute) đặc biệt chỉ rằng lớp đó đã được tiền kiểm tra. Thuộc tính này tăng thêm khoảng 5% kích thước của lớp và sẽ được kiểm tra bởi bộ kiểm tra trên thiết bị di động.

Trên IDE: Tạo tập tin JAR

IDE sẽ tạo một tập tin JAR chưa:

* Tất cả các tập tin \*.class
* Các hình ảnh của ứng dụng. Hiện tại chỉ hỗ trợ tập tin \*.png
* Các tập tin dữ liệu có thể được yêu cầu bởi ứng dụng
* Một tập tin kê khai (manifest.mf) cung cấp mô tả về ứng dụng cho bộ quản lý ứng dụng (application manager) trên thiết bị di động.
* Tập tin JAR được bán hoặc được phân phối đến người dùng đầu cuối.

Sau khi đã gỡ rối và kiểm tra mã lệnh trên trình mô phỏng (simulator), mã lệnh đã sẵn sang được kiểm tra trên điện thoại di động và sau đó được phân phối cho người dùng.

Người dùng: Download ứng dụng về thiết bị di động

Người dùng sau đó download tập tin JAR chứa ứng dụng về thiết bị di động. Trong hầu hết các điện thoại di động, có ba cách để download ứng dụng:

* Kết nối cáp dữ liệu từ PC sang cổng dữ liệu của điện thoại di động: Việc này yêu cầu người dùng phải có tập tin JAR thật sự và phần mềm truyền thông để download ứng dụng sang thiết bị thông qua cáp dữ liệu.
* Cổng hồng ngoại IR (Infra Red) Port: Việc này yêu cầu người dùng phải có tập tin JAR thật sự và phần mềm truyền thông để download ứng dụng sang thiết bị thông qua cổng hồng ngoại.
* OTA (Over the Air): Sử dụng phương thức này, người dùng phải biết địa chỉ URL chỉ đến tập tin JAR.

Trên thiết bị di động:

Bộ tiền kiểm tra: Kiểm tra mã bytecode

Bộ tiền kiểm tra kiểm tra tất cả các lớp đều có một thuộc tính hợp lệ đã được thêm vào bởi bộ tiền kiểm tra trên trạm phát phát triển ứng dụng. Nếu tiến trình tiền kiểm tra thất bại thì ứng dụng sẽ khoogn download về thiết bị di động.

Bộ quản lý ứng dụng: Lưu trữ chương trình

Bộ quản lý ứng dụng trên thiết bị di động sẽ lưu trữ chương trình trên thiết bị di động. Bổ quản lý ứng dụng cũng điều khiển trạng thái của ứng dụng trong thời gian thực thi và có thể tạm dừng ứng dụng khi có cuộc gọi hoặc tin nhắn đến.

Người dùng: Thực thi ứng dụng

Bộ quản lý ứng dụng sẽ chuyển ứng dụng cho KVM để chạy trên thiết bị di động.

KVM: Thực thi mã bytecode khi chương trình chạy.

KVM dịch mã bytecode sang ngôn ngữ máy của thiết bị di động để chạy.

### Tầng CLDC (Connected Limited Device Configuration)

Tầng J2ME kế trên tầng KVM là CLDC hay cấu hình thiết bị kết nối giới hạn. Mục đích của tầng này là cung cấp một tập tối thiểu các thư viện cho phép một ứng dụng Java chạy trên thiết bị di động. Nó cung cấp cơ sở cho tầng hiện trạng, tầng này sẽ chứa nhiều API chuyên biệt hơn.

Các CLDC API được định nghĩa với sự hợp tác với 18 công ty là bộ phận của JCP (Java Community Process). Nhóm này giúp bảo đảm rằng các API được định nghĩa sẽ hữu dụng và thiết thực cho cả nhà phát triển lẫn nhà sản xuất thiết bị di động. Các đặc tả của JCP được gán các số JSR (Java Specification Request). Quy định CLDC phiên bản 1.0 được gán số JSR – 30.

#### CLDC – Connected Limited Device Configuration

Phạm vi: Định nghĩa các thư viện tối thiểu và các API.

Định nghĩa:

* Tương thích ngôn ngữ JVM
* Các thư viện lõi
* I/O
* Mạng
* Bảo mật
* Quốc tế hóa

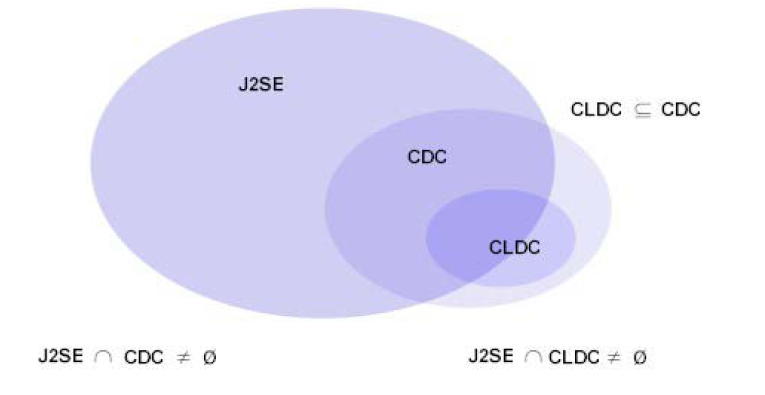
Không định nghĩa:

* Chu kỳ sống ứng dụng
* Giao diện người dùng
* Quản lý sự kiện
* Giao diện ứng dụng và người dùng

Các lớp lõi Java cơ bản, imput/output, mạng, và bảo mật được định nghĩa trong CLDC. Các API hữu dụng hơn như giao diện người dùng và quản lý sự kiện được dành cho hiện trạng MIDP.

J2ME là một phiên bản thu nhỏ của J2SE, sử dụng ít bộ nhớ hơn để nó có thể thích hợp với các thiết bị di động bị giới hạn bộ nhớ. Mục tiêu của J2ME là một tập con 100% tương thích của J2SE.

Hình sau biểu diễn mối liên hệ giữa J2SE và J2ME (CDC, và CLDC).



*Figure 7: J2ME và J2SE*

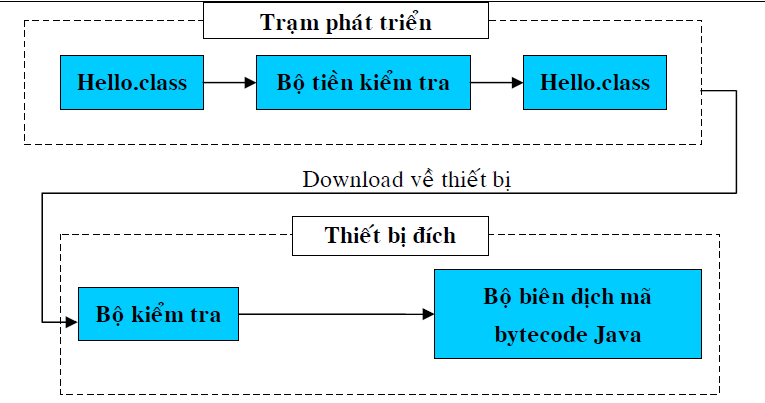
#### Sự khác nhau giữa J2ME và J2SE

Các điểm khác nhau là do một trong hai lý do. Do lớp Java đã bị bỏ đi để giảm kích thước của J2ME hoặc do lớp bị bỏ bởi vì nó ảnh hưởng đến sự an toàn, bảo mật của thiết bị di động hay của các ứng dụng khác trên thiết bị di động (có thể dẫn đến phát triển virus).

Điểm khác biệt chính là không có phép toán số thực. Không có JNI (JavaNative Interface Support) do đó bạn không thể truy xuất các chương trình khác được viết bằng ngôn ngữ của thiết bị (như C hay C++). Tuyến đoạn (thread) được cho phép nhưng không có các nhóm tuyến đoạn (thread group) và các daemon thread.

CLDC định nghĩa một mô hình an toàn, bảo mật được thiết kế để bảo vệ thiết bị di động, KVM, và các ứng dụng khác khỏi các mã phá hoại. Hai bộ phận được định nghĩa bởi CLDC này là bộ tiền kiểm tra và mô hình sandbox.

Hình sau biểu diễn cách mà bộ tiền kiểm tra và bộ kiểm tra làm việc với nhau để kiểm tra mã chương trình Java trước khi chuyển nó cho KVM.

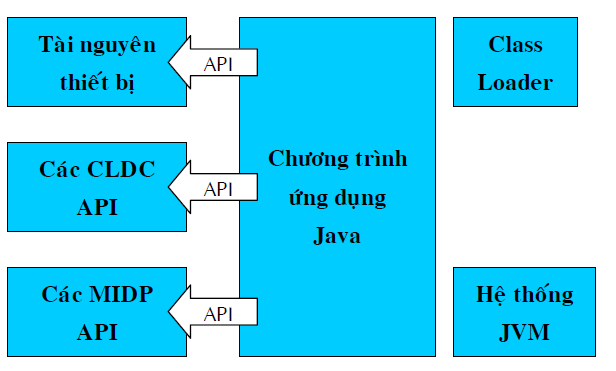


*Figure 8: Bộ tiền kiểmt tra*

Như đã đề cập trước đây, các tập tin lớp được gán nhãn bằng một thuộc tính trên máy trạm của nhà phát triển. Thuộc tính này sau đó được kiểm tra bởi bộ tiền kiểm tra trước khi mã chương trình được giao cho KVM hay bộ biên dịch mã bytecode.

Một bộ phận khác của bảo mật trong CLDe là mô hình sandbox.

Hình sau biểu diễn khái niệm mô hình sandbox:



*Figure 9: Mô hình sandbox*

Hình trên cho thấy ứng dụng J2ME đặt trong một sandbox có nghĩa là nó bị giới hạn truy xuất đến tài nguyên của thiết bị và không được truy xuất đến máy ảo Java hay bộ nạp chương trình. Ứng dụng được truy xuất đến các API của CLDC và MIDP. Ứng dụng được truy xuất tài nguyên của thiết bị di động (các cổng, âm thanh, bộ rung, các báo hiệu, …) chỉ khi nhà sản xuất điện thoại di động cung cấp các API tương ứng. Tuy nhiên, các API này không phải là một phần của J2ME.

Thế hệ kế tiếp của CLDC là đặc tả JSR – 139 và được gọi là CLDC thế hệ kế tiếp (Next Generation). Nó sẽ nhắm đến các vấn đề như nâng cao việc quản lý lỗi và có thể phép toán số thực.

### Sự cần thiết của J2ME

Thế giới của các thiết bị di động và các thiết bị “sub-PC” không có các đặc tính giống như trong lĩnh vực PC và server.

Ngoài ra, không phải mọi thiết bị trong lĩnh vực này đều cùng làm một việc. Sự khác nhau về thiết kế và mục đích giữa PDA, điện thoại, và máy nhắn tin là rất đáng kể.

Bất kể nó mang lại sự đổi mới gì cho thị trường, thì tính đa dạng của các thiết bị này là một ác mộng đối với các lập trình viên. Nếu tôi muốn xây dựng một ứng dụng cho điện thoại di động, tôi có phải viết mã lại, xây dựng lại, và kiểm tra lại cho mọi thiết bị hay không? Nếu tôi muốn xây dựng một client có kết nối mạng, tôi phải xét đến các công nghệ kết nối nào? V.v.

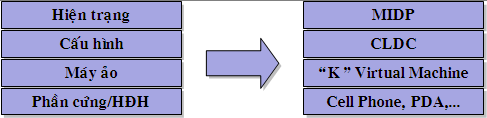
J2ME ra đời nhằm mục đích chính là thiết lập một chuẩn đơn mà thông qua đó các nhà phát triển có thể tạo nên các phần mềm có tính khả chuyển (portable) cho các thiết bị micro. Ngôn ngữ Java là sự lựa chọn đương nhiên cho lĩnh vực này, bởi vì về cơ bản nó đã hướng nhiều về tính khả chuyển. Bằng cách này, Sun đã đảm nhận bài toán lớn về tính đa dạng của thiết bị ở một mức tổng quát, do đó các nhà phát triển không phải quan tâm đến vấn đề này nữa. Nếu mọi nhà cung cấp PDA, điện thoại và máy nhắn tin đều thực hiện J2ME cho thiết bị của họ, thì chúng ta có khả năng viết chương trình “viết một lần, chạy mọi nơi” (write once, run anywhere) trong lĩnh vực micro, cũng giống như ta đã làm quen với khái niệm này ở các hệ thống máy lớn.

### Hiện trạng thiết bị thông tin di động (Mobile Information Device Profile)

Mặc dù không phải chỉ có một hướng kiến trúc J2ME, nhưng các thiết bị di động không dây dường như dần dần càng quan tâm đến J2ME. Bao gồm:

* Điện thoại di động
* Trợ tá cá nhân số (Personal Digital Assistant- PDA)
* Máy nhắn tin
* Thiết bị đọc sách điện tử
* Các thiết bị point-of-sale

J2ME được tổ chức thành các mức, mỗi mức xác định một định nghĩa tăng dần của các thiết bị đích. Có nhiều lựa chọn kiến trúc tồn tại ở mỗi mức, và ràng buộc tùy chọn ở các mức cao hơn. Lập trình viên chỉ cần quan tâm đến hiện trạng (profile), định nghĩa các API, các nhà thực hiện J2ME cho thiết bị cần tập trung đến mức VM (Virtual Machine).



*Figure 10: Các mức tổ chức J2ME*

Các đặc tả cho các thiết bị không dây là Connected Limited Device Configuration hay CLDC, và Mobile Information Device Profile hay MIDP. MIDP định nghĩa các đặc tính tối thiểu của thiết bị như sau:

Bộ nhớ không dây bay hơi có dung lượng 128K (nghĩa là, bộ nhớ có trạng thái được giữ lại khi thiết bị đã tắt) dành cho các thành phần MIDP, bao gồm KVM, Core API và chương trình MIDP.

8K bộ nhớ không bay hơi dành cho dữ liệu bền vững của ứng dụng

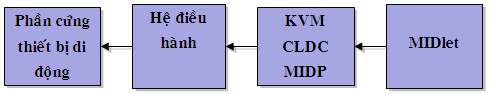
32K bộ nhớ bay hơi cho bộ nhớ của chương trình.

Màn hình hiển thị ít nhất là 96x54 pixel, có thể chỉ là một bit màu hay hỗ trợ nhiều màu hoặc màu mức xám.

Cơ chế nhập liệu hỗ trợ ít nhất một bộ phím số, hoặc một màn hình cảm ứng có khả năng cấu hình hỗ trợ nhập liệu số.

Khả năng kết nối mạng không dây hai chiều, với băng thông hạn chế và thông thường là không liên tục

Như vậy các thiết bị hỗ trợ MIDP cung cấp một nền tảng chuẩn cho các phần mềm Java:



*Figure 11: Triển khai hệ thống J2ME*

### Các kiểu ứng dụng MIDP

Các ứng dụng MIDP được gọi là các MIDlet. Hầu hết các MIDlet đều ở một trong hai dạng sau:

##### Ứng dụng đơn (standalone application) được nạp hoàn toàn vào thiết bị và có thể chạy bất kỳ lúc nào thiết bị được mở, không yêu cầu tài nguyên bên ngoài. Loại này bao gồm:

* Các ứng dụng PDA và ứng dụng organizer như sổ địa chỉ, danh sách công việc và lịch hẹn.
* Các công cụ đơn giản như máy làm tính (calculator)
* Trò chơi

##### Ứng dụng nối mạng (networked application) được chia thành ít nhất hai thành phần, một thành phần là client được triển khai trên thiết bị di động. Thành phần này sẽ ít được dùng nếu không có kết nối ít nhất một server trên hệ thống. Server thường là được đặt trong môi trường J2EE, và phục vụ bằng Web hoặc các giao thức Internet khác.

Ở đây, ta hãy kỹ thuật ngữ “client”. Ta không gọi một MIDlet là một client chỉ đơn giản là vì nó sử dụng kết nối mạng MIDP và liên lạc đến các thành phần khác. Câu hỏi là phần luận lý lõi của ứng dụng đặt ở đâu? MIDlet có đảm nhận hầu hết việc “suy nghĩ” và chỉ quan tâm đến mạng hay không? Đó không phải là client, thật vậy – ít nhất là không theo đúng nghĩa trong ngữ cảnh của hệ thống enterprise. Một client là một MIDlet dựa vào một server đẻ suy nghĩ, lưu trữ, tải, xử lý, hay nói cách khác là làm việc thay cho nó.

CHƯƠNG 3

# GOOGLE MAP API

**Google Maps** (thời gian trước còn gọi là **Google Local**) là một dịch vụ ứng dụng và công nghệ bản đồ trực tuyến trên web miễn phí được cung cấp bởi Google và hỗ trợ nhiều dịch vụ dựa vào bản đồ như Google Ride Finder và một số có thể dùng để nhúng vào các trang web của bên thứ ba thông qua Google Maps API. Nó cho phép thấy bản đồ đường sá, đường đi cho xe đạp, cho người đi bộ (những đường đi ngắn hơn 6.2 dặm) và xe hơi, và những địa điểm kinh doanh trong khu vực cũng như khắp nơi trên thế giới.

Ngày nay, với những người sử dụng Internet, không ai còn xa lạ gì với Google, đầu tiên nó được biết đến là một cỗ máy tìm kiếm thông tin tối ưu trên mạng Internet. Google nhanh chóng được biết đến bởi tất cả những người dùng Internet.

* ***Vậy Google là gì?***

Tên Google là một cách chơi chữ từ từ "googol" tạo ra bởi Milton Sirrota. Một googol có nghĩa là một con số đại diện bằng số 1 với 100 số không phía sau. Nó là một số rất lớn. Trong thực tế không có cái gì có số lượng lớn đến như thế (bao gồm cả bụi, cát, nguyên tử). Trang tìm kiếm Google sử dụng thuật ngữ này để chỉ nhiệm vụ rất lớn của trang web là đáp ứng yêu cầu to lớn của cả thế giới về tìm kiếm thông tin làm cho thông tin phổ dụng trên toàn cầu.

Từ cái tên gọi như vậy cho thấy tham vọng của những người khởi dựng lên Google là như thế nào. Larry E. Page và Sergey Brin - những người sáng lập giàu tham vọng của Google - từ lâu đã tuyên bố sứ mệnh của Google là "sắp xếp và quản lý thông tin toàn cầu". Hiện tại, ai cũng công nhận là sứ mệnh của Google đang được thực hiện rất tốt. Kho dữ liệu của Google gồm trên 6 tỷ mục thông tin, bao gồm 4,28 tỷ trang web, 880 triệu hình ảnh và 845 triệu thông điệp Internet. Trong ba năm qua, Google đã đi từ chỗ có 100 triệu lượt tra cứu một ngày đến hơn 200 triệu lượt tra cứu một ngày, trong đó chỉ có 1/3 lượt tra cứu là từ nước Mỹ, số lượt tra cứu còn lại là từ 88 quốc gia trên thế giới.

Một mảng thông tin đang tăng nhanh gần đây, rất tiện ích đối với học sinh sinh viên, các nhà nghiên cứu là các trang liên quan đến sách, bao gồm các chương đầu, phần phê bình, tham khảo. Hệ thống thông tin này được Google truy xuất qua dịch vụ Google Print mà họ đang cho vận hành thử nghiệm. Google đang dần số hóa các thư viện trên thế giới và chuẩn bị cung cấp dịch vụ băng thông rộng không dây tới hàng triệu người trên trái đất.

Theo Peter Norvig, Giám đốc phụ trách chất lượng tìm kiếm của Google, cùng với việc tăng cường link thêm những trang web mới, Google còn liên tục cải tiến các thuật toán xếp hạng của mình để đưa ra những kết quả tìm kiếm ngày càng gần với nội dung mà người truy cập yêu cầu.



*Figure 15: Bản đồ Google*

Cùng với việc phát triển không ngừng của mình, Google liên tục đưa ra các sản phẩm dịch vụ mới trên mạng Internet và một trong những dịch vụ lớn đó là bản đồ thông tin của toàn thế giới Google Map.

Google Map là một dịch vụ của Google cung cấp công nghệ bản đồ mạnh, thân thiện với người dùng và các thông tin của doanh nghiệp địa phương bao gồm địa điểm doanh nghiệp, thông tin liên hệ, và chỉ đường.

Ngoài ra một đặc tính quan trọng của Google Map là ngoài việc cho phép người dùng tra cứu, xem thông tin trên bản đồ nó còn cho phép mọi người có thể nhúng bản đồ lên một trang web bất kỳ của mình chỉ bằng cách tạo một tài khoản trong dịch vụ Google hoàn toàn miễn phí.

Bên cạnh đó Google cũng cung cấp một thư viện các API bằng Javascript cho phép người khác sau khi nhúng bản đồ Google Map lên trang web của mình có thể khai thác và thực hiện các công việc phát triển liên quan.

CHƯƠNG 4

# GPS VÀ VẤN ĐỀ ĐỊNH VỊ TỌA ĐỘ

## Hệ thống định vị toàn cầu – GPS là gì?

**Hệ thống định vị toàn cầu** (*Global Positioning System* - **GPS**) là hệ thống xác định vị trí dựa trên vị trí của các vệ tinh nhân tạo. Trong cùng một thời điểm, ở một vị trí trên mặt đất nếu xác định được khoảng cách đến ba vệ tinh (tối thiểu) thì sẽ tính được tọa độ của vị trí đó.

GPS bao gồm các vệ tinh bay trên quỹ đạo, thu thập thông tin toàn cầu và được xử lý bởi các trạm điều khiển trên mặt đất. Ngày nay, khó hình dung rằng có một máy bay, một con tàu hay phương tiện thám hiểm trên bộ nào lại không lắp đặt thiết bị nhận tín hiệu từ vệ tinh.



*Figure 16: Minh họa các vệ tinh GPS và quỹ đạo của nó*

GPS được thiết kế và bảo quản bởi Bộ Quốc phòng Hoa Kỳ, nhưng chính phủ Hoa Kỳ cho phép mọi người trên thế giới sử dụng nó miễn phí, bất kể quốc tịch.

Các nước trong Liên minh châu Âu đang xây dựng Hệ thống định vị Galileo, có tính năng giống như GPS của Hoa Kỳ, dự tính sẽ bắt đầu hoạt động năm 2011-2012

## Lịch sử phát triển GPS

Năm 1978, nhằm mục đích thu thập các thông tin về tọa độ (vĩ độ và kinh độ), độ cao và tốc độ của các cuộc hành quân, hướng dẫn cho pháo binh và các hạm đội, Bộ Quốc phòng Mỹ đã phóng lên quỹ đạo trái đất 24 vệ tinh. Số lượng vệ tinh GPS theo số liệu năm 1998 là 28 vệ tinh và hiện nay là 31 vệ tinh (số liệu 2008).

Những vệ tinh trị giá nhiều tỷ USD này bay phía trên trái đất ở độ cao 19.200 km, với tốc độ chừng 11.200 km/h, có nhiệm vụ truyền đi các tín hiệu radio tần số thấp tới các thiết bị thu nhận. Từ những năm đầu thập kỷ 80, các nhà sản xuất lớn chú ý nhiều hơn đến đối tượng sử dụng tư nhân. Trên các xe hơi hạng sang, những thiết bị trợ giúp cá nhân kỹ thuật số PDA (Personal Digital Assistant) như Ipaq của hãng Compaq, được coi là một trang bị tiêu chuẩn, thể hiện giá trị của chủ sở hữu.

Trong số 24 vệ tinh của Bộ quốc phòng Mỹ nói trên, chỉ có 21 thực sự hoạt động, 3 vệ tinh còn lại là hệ thống hỗ trợ. Tín hiệu radio được truyền đi thường không đủ mạnh để thâm nhập vào các tòa nhà kiên cố, các hầm ngầm và hay tới các địa điểm dưới nước. Ngoài ra nó còn đòi hỏi tối thiểu 4 vệ tinh để đưa ra được thông tin chính xác về vị trí (bao gồm cả độ cao) và tốc độ của một vật. Vì hoạt động trên quỹ đạo, các vệ tinh đảm bảo cung cấp vị trí tại bất kỳ điểm nào trên trái đất.

GPS ban đầu chỉ dành cho các mục đích quân sự, nhưng từ năm 1980 chính phủ Mỹ cho phép sử dụng dân sự. GPS hoạt động trong mọi điều kiện thời tiết, mọi nơi trên Trái Đất, 24 giờ một ngày. Không mất phí thuê bao hoặc mất tiền trả cho việc thiết lập sử dụng GPS.

## Phân loại

Hệ thống định vị toàn cầu của Mỹ là hệ dẫn đường dựa trên một mạng lưới các vệ tinh được Bộ Quốc phòng Hoa Kỳ đặt trên quỹ đạo không gian.

Các hệ thống dẫn đường truyền thống hoạt động dựa trên các trạm phát tín hiệu vô tuyến điện. Được biết nhiều nhất là các hệ thống có tên gọi LORAN – (*LOng RAnge Navigation*) – hoạt động ở giải tần 90-100 kHz chủ yếu dùng cho hàng hải, hay TACAN – (*TACtical Air Navigation*) – dùng cho quân đội Mỹ và biến thể với độ chính xác thấp VOR/DME – VHF (*Omnidirectional Range/Distance Measuring Equipment*) – dùng cho hàng không dân dụng.

Gần như đồng thời với lúc Mỹ phát triển GPS, Liên Xô cũng phát triển một hệ thống tương tự với tên gọi GLONASS. Hiện nay Liên minh Châu Âu đang phát triển hệ dẫn đường vệ tinh của mình mang tên Galileo.

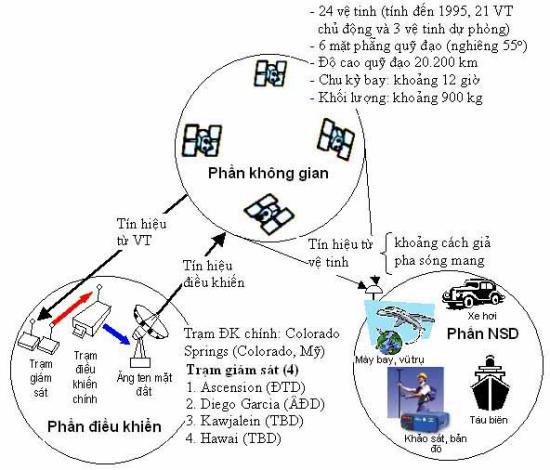
Chú ý rằng cả GPS và GLONAS đều được phát triển trước hết cho mục đích quân sự. Nên mặc dù chúng có cho dùng dân sự nhưng không hệ nào đưa ra sự đảm bảo tồn tại liên tục và độ chính xác. Vì thế chúng không thỏa mãn được những yêu cầu an toàn cho dẫn đường dân sự hàng không và hàng hải, đặc biệt là tại những vùng và tại những thời điểm có hoạt động quân sự của những quốc gia sở hữu các hệ thống đó. Chỉ có hệ thống dẫn đường vệ tinh châu Âu Galileo (đang được xây dựng) ngay từ đầu đã đặt mục tiêu đáp ứng các yêu cầu nghiêm ngặt của dẫn đường và định vị dân sự.

## Sự hoạt động của GPS và tín hiệu GPS

### Sự hoạt động của GPS

Các vệ tinh GPS bay vòng quanh Trái Đất hai lần trong một ngày theo một quỹ đạo rất chính xác và phát tín hiệu có thông tin xuống Trái Đất. Các máy thu GPS nhận thông tin này và bằng phép tính lượng giác tính được chính xác vị trí của người dùng. Về bản chất máy thu GPS so sánh thời gian tín hiệu được phát đi từ vệ tinh với thời gian nhận được chúng. Sai lệch về thời gian cho biết máy thu GPS ở cách vệ tinh bao xa. Rồi với nhiều khoảng cách đo được tới nhiều vệ tinh máy thu có thể tính được vị trí của người dùng và hiển thị lên bản đồ điện tử của máy.

Máy thu phải khóa được với tín hiệu của ít nhất ba quả vệ tinh để tính ra vị trí hai chiều (kinh độ và vĩ độ) và để theo dõi được chuyển động. Với bốn hay nhiều hơn số quả vệ tinh trong tầm nhìn thì máy thu có thể tính được vị trí ba chiều (kinh độ, vĩ độ và độ cao). Một khi vị trí người dùng đã tính được thì máy thu GPS có thể tính các thông tin khác, như tốc độ, hướng chuyển động, bám sát di chuyển, khoảng hành trình, khoảng cách tới điểm đến, thời gian Mặt Trời mọc, lặn và nhiều thứ khác nữa.



*Figure 17: Minh họa sự hoạt động của hệ thống GPS*

Việc định vị dựa trên hệ thống GPS chính là việc đo khoảng cách từ vệ tinh đến máy thu (nếu máy thu thu được thông tin từ 3 vệ tinh, nó sẽ tính ra được vị trí của nó trên địa cầu).Việc đo khoảng này thực tế chính là việc đo thời gian từ lúc vệ tinh phát tín hiệu đến lúc máy thu thu được tín hiệu và vì vận tốc sóng điện từ bằng vận tốc ánh sáng nên từ đó tính ra được khoảng cách.

Để có thể xác định được khoảng thời gian từ lúc phát tín hiệu đến lúc máy thu thu được tín hiệu, trên mỗi vệ tinh GPS bay quanh trái đất có gắn một đồng hồ nguyên tử và được đồng bộ với nhau. Đồng thời trên các thiết bị thu (GPS Receiver) cũng có một đồng hồ, đồng hồ này luôn được reset một cách liên tục và khi tiếp nhận thông tin từ các vệ tinh GPS, chúng có thể tính được khoảng thời gian sai lệch từ khi gởi đến khi nhận tín hiệu, từ đó, dựa vào vận tốc và thời gian tính được, chúng ta có được khoảng cách từ vị trí GPS Receiver đến vệ tinh GPS.

Khi có khoảng cách từ một vệ tinh thì vị trí của máy thu chính là tập hợp các điểm trên mặt cầu có bán kính là khoảng cách tính được và tâm là vệ tinh phát. Với bốn khoảng cách thu được, ta sẽ loại bỏ nghiệm và còn lại một nghiệm duy nhất chính là vị trí máy thu.

Tuy nhiên, cách tính trên là áp dụng với điều kiện trái đất là hình cầu. Thực tế thì trái đất lồi lõm và không tròn đều nên sẽ gây ra sai số khi tính toán, vậy nên người ta đã xây dựng các trạm vi phân GPS (gọi là DGPS) để hiệu chỉnh các thông tin thu được từ vệ tinh kết hợp với các thông số trắc địa thực tế tại khu vực đó.

### Vệ tinh GPS xác định một điểm trên trái đất như thế nào?

Theo như phần giới thiệu về hoạt động của GPS, ta biết được rằng GPS xác định một ví trí bất kỳ trên trái đất dựa trên khoảng cách giữa máy thu GPS và các vệ tinh.

Để có thể hiểu hơn về cách thức xác định một vị trí trên mặt đất của các vệ tinh GPS ta có thể tìm hiểu thêm về các ví dụ như sau:

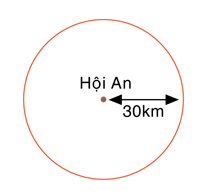
**Ví dụ 1: Xét trường hợp xác định 1 vị trí trên mặt phẳng 2D**

Giả sử một người là A đang đứng ở một nơi nào đó trên bề mặt trái đất và không thể xác định được mình đang ở đâu, anh ta có thể hỏi một người khác

Tôi đang ở đâu?

Anh đang ở cách Hội An 30 km. – Người thứ nhất trả lời

Sau khi nhận được câu trả lời, rõ ràng người A lúc này chỉ có thể biết được rằng mình đang ở một điểm nào đó trên một đường tròn mà có tâm là Hội An và bán kính là 30 km.



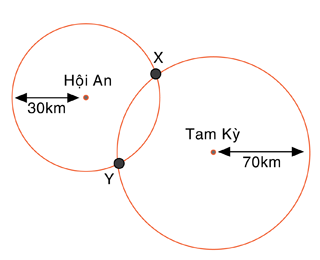
*Figure 18: Người A đang ở cách Hội An 30 Km*

Tiếp tục, người A sẽ hỏi người thứ 2 cũng với câu hỏi ban đầu:

Tôi đang ở đâu?

Anh đang ở cách Tam Kỳ 70 Km – Người thứ 2 trả lời.

Với thông tin có được từ người thứ 2, người A có thể biết được rằng mình đang ở một điểm nào đó trên một đường tròn có tâm là Tam Kỳ và bán kính là 70 km. Kết hợp với thông tin có được từ người thứ nhất, rõ ràng người A lúc này đang ở một trong hai vị trí cắt nhau bởi hai đường tròn có tâm là Hội An, bán kính 30 km và một đường tròn khác có tâm là Tam Kỳ, bán kính 70 km.



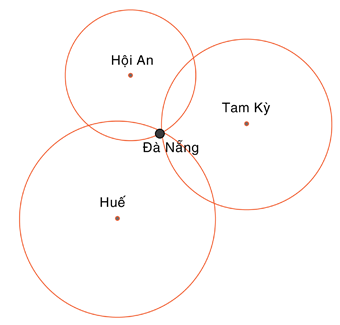
*Figure 19: Người A đang ở một trong hai điểm X và Y*

Và để tiếp tục xác định được chính xác vị trí của mình, người A tiếp tục hỏi người thứ 3

Tôi đang ở đâu?

Anh đang ở cách Huế 100 km – người thứ 3 trả lời.

Tương tự như trên, người A cũng có thể biết được rằng mình đang ở một điểm nào đó trên một đường tròn có tâm là Huế và bán kính là 100 km, và đồng thời cũng biết chắc chắn rằng đường tròn này sẽ cắt một trong hai điểm giao nhau của hai đường tròn đã xác định từ những thông tin trước và từ đó người A có thể biết chính xác rằng, anh ta đang đứng ở Đà Nẵng.

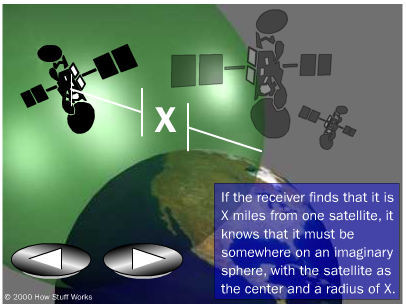


*Figure 20: Xác định người A đang ở Đà Nẵng thông qua đường tròn thứ 3*

Qua ví dụ trên, ta có thể hình dung được rằng cách mà các vệ tinh GPS dựa vào vị trí của mình xác định một vị trí trên bề mặt trái đất hoàn toàn tương tự, tuy nhiên vì các vệ tinh GPS nằm trong không gian cho nên chúng ta cần hình dung lại ví dụ trên theo trong không gian ba chiều (3D). Tương tự như ví dụ ở trên, ta có thể hình dung ra mỗi vệ tinh GPS sẽ là tâm của một hình cầu có bán kính bằng đúng khoảng cách của chúng đến điểm cần xác định. Tuy nhiên để có thể xác định một điểm trong không gian, ta phải cần đến bốn hình cầu cắt nhau thay vì là ba hình tròn như trong không gian hai chiều.

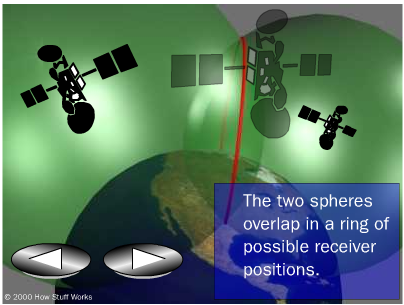
Ta có thể nghiên cứu rõ hơn về vấn đề này như sau:

Một vị trí X nào đó sau khi tiếp nhận tín hiệu từ một vệ tinh A, nghĩa là ta xác định được khoảng cách từ nó đến vệ tinh A, thì tương tự như trên, ta có thể biết được rằng X đang ở một điểm nào đó trên bề mặt hình cầu có tâm là A.



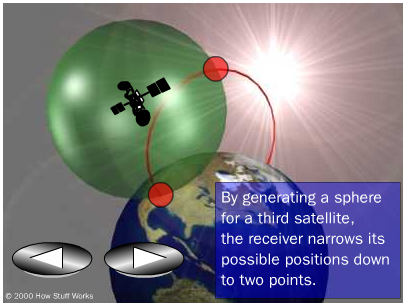
*Figure 21: Khi X nhận tín hiệu từ một vệ tinh*

Tiếp đến, **X** sẽ nhận tín hiệu từ một vệ tinh **B**, ở đây **X** cũng sẽ nằm trên một điểm ở bề mặt hình cầu có tâm là **B**, và như ở ví dụ trên cùng với hình dung về mặt hình học không gian, hai hình cầu có tâm là **A** và **B** sẽ cắt nhau và cho chúng ta một hình tròn, đến đây ta tạm gọi nó là đường tròn **K**. Lúc này ta cũng có thể xác định được rằng điểm **X** sẽ nằm ở một điểm bất kỳ trên đường tròn **K**.



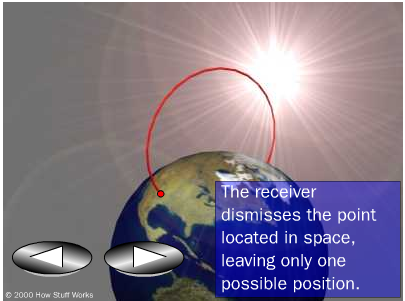
*Figure 22: Khi X nhận tín hiệu từ 2 vệ tinh*

Chúng ta xác định tiếp khoảng cách của điểm **X** đối với vệ tinh **C**, ta sẽ tiếp tục có một hình cầu thứ ba và điểm **X** sẽ là một điểm trên bề mặt của nó, đồng thời hình cầu thứ ba này sẽ cắt đường tròn **K** có chứa điểm **X** tại hai điểm, **X** lúc này sẽ nằm một trong hai điểm đó.



*Figure 23: Điểm X khi nhận tín hiệu từ vệ tinh thứ 3*

Như ở phần trên ta đã nêu, để có thể xác định được tọa độ của một điểm ta cần tối thiểu cùng lúc ba tín hiệu từ ba vệ tinh khác nhau, như vậy, sau khi có đủ ba tín hiệu vệ tinh, làm thế nào để ta có thể xác định được điểm **X** nằm ở đâu trên hai vị trí thu được? Lúc này, trái đất của chúng ta sẽ đóng vai trò như là một hình cầu thứ tư (xác định một điểm trong không gian cần có bốn hình cầu giao nhau), và hẳn nhiên một điều, điểm X cũng sẽ là một điểm bất kỳ trên bề mặt trái đất. Đến đây ta có thể dễ dàng hình dung rằng, từ hai điểm xác định ở trên, điểm **X** sẽ là điểm giao nhau giữa trái đất với hai điểm ở trên.



*Figure 24: Điểm X được xác định*

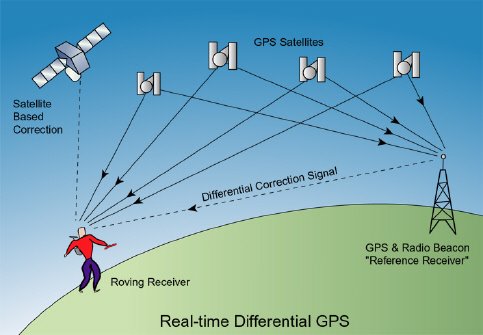
### Các thành phần của GPS

Hệ thống định vị toàn cầu GPS bao gồm có 3 phần:

* **Phần không gian:**

Theo thiết kế ban đầu, chùm vệ tinh GPS sẽ có 21 vệ tinh hoạt động và 3 vệ tinh dự trữ có thể đưa vào hoạt động bất cứ lúc nào nhờ điều khiển từ mặt đất. Tuy nhiên, theo số liệu của năm 2008, hiện nay trên không gian đang có 31 vệ tinh GPS nằm trên các quỹ đạo xoay quanh trái đất. Chúng cách mặt đất 12 nghìn dặm (~ 20200 km), chu kỳ khoảng12h (trong vòng 24h chúng bay vòng quay trái đất 2 lần với vận tốc 7 nghìn dặm/h) và chia làm 6 mặt phẳng quĩ đạo, bán kính quĩ đạo 26560 km, góc nghiêng của mặt phẳng quĩ đạo với mặt phẳng xích đạo là 550 . Các điểm nút lên và nút xuống của các mặt phẳng quĩ đạo lệch nhau 600. Các vệ tinh có một hệ thống đẩy để duy trì vị trí trên quĩ đạo và điều chỉnh cho ổn định. Như vậy, bất cứ thời điểm nào từ an-ten máy thu cũngcó thể nhìn thấy khoảng 10 - 11 vệ tinh cùng lúc cho phép máy thu lựa chọn được các vệ tinh có đặc tính hình học không gian tốt nhất để xác định vị trí. Thông thường các máy thu GPS trên mặt đất cần nhìn thấy tối thiểu bốn vệ tinh vào bất kỳ thời điểm nào để xác định được chính xác vị trí.

Các vệ tinh được cung cấp bằng năng lượng Mặt Trời. Chúng có các nguồn pin dự phòng để duy trì hoạt động khi chạy khuất vào vùng không có ánh sáng Mặt Trời. Các tên lửa nhỏ gắn ở mỗi quả vệ tinh giữ chúng bay đúng quỹ đạo đã định.



Hình : Sự hoạt động của các thành phần GPS

* **Phần kiểm soát:**

Mục đích trong phần này là kiểm soát vệ tinh đi đúng hướng theo quỹ đạo và thông tin thời gian chính xác. Có tất cả năm trạm kiểm soát được đặt rải rác trên trái đất. Bốn trạm kiểm soát hoạt động một cách tự động, và một trạm kiểm soát là trung tâm. Bốn trạm này nhận tín hiệu liên tục từ những vệ tinh và gửi các thông tin này đến trạm kiểm soát trung tâm. Tại trạm kiểm soát trung tâm, nó sẽ sửa lại dữ liệu cho đúng và kết hợp với hai ăngten khác để gửi lại thông tin cho các vệ tinh.

* **Phần sử dụng:**

Phần sử dụng là thiết bị nhận tín hiệu vệ tinh GPS và người sử dụng thiết bị này thường được gọi là GPS Receiver.

Dưới đây là một số thông tin đáng chú ý về các vệ tinh GPS (còn gọi là NAVSTAR, tên gọi chính thức của Bộ Quốc phòng Mỹ cho GPS):

Vệ tinh GPS đầu tiên được phóng năm 1978.

Hoàn chỉnh đầy đủ 24 vệ tinh vào năm 1994.

Đến năm 1998 số lượng vệ tinh là 28

Năm 2008, số lượng vệ tinh là 31

Mỗi vệ tinh được làm để hoạt động tối thiểu là 10 năm.

Vệ tinh GPS có trọng lượng khoảng 1500 kg và dài khoảng 17 bộ (5 m) với các tấm năng lượng Mặt Trời mở (có độ rộng 7 m²).

Công suất phát bằng hoặc dưới 50 watts.

### Tín hiệu GPS

Các vệ tinh GPS phát hai tín hiệu vô tuyến công suất thấp giải L1 và L2. (Giải L là phần sóng cực ngắn của phổ điện từ trải rộng từ 0,39 tới 1,55 GHz). GPS dân sự dùng tần số L1 1575.42 MHz trong giải UHF. Tín hiệu truyền trực thị, có nghĩa là chúng sẽ xuyên qua mây, thuỷ tinh và nhựa nhưng không qua phần lớn các đối tượng cứng như núi và nhà.

Tần số của L1 = 1575,42 MHz và L2 = 1227,6MHz. Mỗi máy phát của vệ tinh đều có bộ dao động tạo tần số 10.23 MHz ổn định nhờ đồng hồ nguyên tử Cesi. Tần số này được nhân 154 lần để được L1 (có bước sóng λ  = 19 cm), nhân 120 lần để được L2 (λ = 24 cm).

Một đài phát thanh FM thường cần có công suất chừng 100.000 watt để phát sóng, nhưng một vệ tinh định vị toàn cầu chỉ đòi hỏi 20-50 watt để đưa tín hiệu đi xa 19.200 km. Tần số L1 chứa đựng 2 tín hiệu số (mã hoá bằng kỹ thuật số) gọi là “mã giả ngẫu nhiên - pseudo random”, được gọi là Protected (P)-code và Coarse/Acquisition (C/A)-code. Mỗi một vệ tinh có một mã truyền dẫn nhất định, cho phép máy thu GPS nhận dạng được tín hiệu. Mục đích của các mã tín hiệu này là để tính toán khoảng cách từ vệ tinh đến máy thu GPS.

Mã P (Pseudo-random) là một dãy các phần tử (chip) gồm các số 0 và 1, tần số phần tử là 10,23 Mbit, chiều dài mỗi phần tử là 99,75 ns (nano second). Chiều dài một dãy phần tử ở mã P là một tuần. Hàng tuần vào 00h00 UTC thứ bảy/chủ nhật lại bắt đầu một dãy/chuỗi mới. Như vậy chuỗi phần tử của mã P rất dài và không lặp lại gây khó khăn cho việc đồng pha và xác định thời gian truyền sóng, nhưng nó có ưu điểm là nâng cao được độ chính xác và có thể hạn chế việc sử dụng mã P chỉ cho một số đối tượng người sủ dụng nhất định (các mục đích quân sự). Mã P cung cấp vị trí với độ chính xác cao 10-16m (Chế độ định vị chính xác Precise Positioning Service - PPS). chỉ có các máy thu đặc biệt mới có thể thu được tín hiệu mã P và từ đó có được vị trí có độ chính xác cao. Tín hiệu mã P được phát trên cả hai tần số L1 và L2. Như vậy mã P chính là nhằm bảo vệ thông tin khỏi những sự truy nhập trái phép. Tuy nhiên, mục đích chính của các tín hiệu mã hóa là nhằm tính toán thời gian cần thiết để thông tin truyền từ vệ tinh tới một thiết bị thu nhận trên mặt đất. Sau đó, khoảng cách giữa hai bên được tính bằng cách nhân thời gian cần thiết để tín hiệu đến nơi với tốc độ của ánh sáng là 300.000 km/giây (khoảng cách = vận tốc x thời gian).

Mã C/A (Coarse Acquisition) là một dãy các phần tử, tần số phần tử là 1,023 Mbit, chiều dài mỗi phần tử là 0,9975 microsecond (gấp 10 lần so với mã P), chiều dài dãy là 1 ms và dãy (chuỗi) phần tử được lặp lại sau mỗi ms đó. Mã C/A cung cấp vị trí với độ chính xác kém hơn mã P, khoảng dưới 50m (Chế độ định vị tiêu chuẩn Standard Positioning Service - SPS). Nhưng thực tế hàng hải coi như độ chính xác là nhỏ hơn 100 m. Chế độ định vị này áp dụng cho mọi loại máy thu dân dụng. Mã C/A chỉ phát trên tần số L1.

Cả hai mã P và C/A đều chứa các bản tin vệ tinh (satellite message) có tốc độ dữ liệu là 50 bit/s. Như vậy, các máy thu dân dụng sử dụng chế độ định vị tiêu chuẩn bằng mã C/A sẽ có độ chính xác kém so với chế độ định vị chính xác bằng mã P về một số nguyên nhân sau:

Chuỗi tín hiệu của mã P rất dài và không lặp lại gây khó khăn cho việc đồng pha để xác định thời gian truyền sóng, các máy thu thông thường cũng không có khả năng tạo chuỗi mô hình giống như chuỗi thật, nên không thể thu được mã P.

Mã P được phát trên hai tần số, do các tần số khác nhau nên sự khúc xạ của sóng khi qua các tầng khí quyển của Trái đất. Máy thu quân sự có thể thu được cả hai tần số này, so sánh kết quả và tính toán được khoảng cách đúng từ vệ tinh đến máy thu. Trong khi máy thu thông thường chỉ thu tần số L1 nên không loại trừ được sai số khúc xạ nói trên.

Mã tín hiệu C/A chịu một sai số do các bản tin vệ tinh bị cố ý làm sai lệch đi, máy thu không thể xác định chính xác thời gian truyền sóng từ vệ tinh đến máy thu, do đó độ chính xác của vị trí bị suy giảm đi.

Đồng hồ của máy thu đặc biệt là loại đồng hồ nguyên tử có độ chính xác rất cao, cao hơn nhiều so với đồng hồ điện tử của máy thu thông thường. Chiều dài chip của mã P chỉ bằng 1/10 so với mã C/A, do đó nó có thể đo thời gian truyền sóng với độ chính xác cao hơn nhiều so với mã C/A.

Để cải thiện độ chính xác, các máy thu thông thường có thể sử dụng kỹ thuật vi sai GPS (Differential GPS). Chế độ này cho phép xác định vị trí với độ chính xác dưới 10 m.

Trên thực tế, tín hiệu GPS chứa ba mẫu thông tin khác nhau: mã giả ngẫu nhiên, dữ liệu thiên văn và dữ liệu lịch. Mã giả ngẫu nhiên đơn giản chỉ là mã định danh để xác định được quả vệ tinh nào là phát thông tin nào. Có thể nhìn số hiệu của các quả vệ tinh trên trang vệ tinh của máy thu Garmin để biết nó nhận được tín hiệu của quả nào.

Dữ liệu thiên văn cho máy thu GPS biết quả vệ tinh ở đâu trên quỹ đạo ở mỗi thời điểm trong ngày. Mỗi quả vệ tinh phát dữ liệu thiên văn chỉ ra thông tin quỹ đạo cho vệ tinh đó và mỗi vệ tinh khác trong hệ thống.

Dữ liệu lịch được phát đều đặn bởi mỗi quả vệ tinh, chứa thông tin quan trọng về trạng thái của vệ tinh (lành mạnh hay không), ngày giờ hiện tại. Phần này của tín hiệu là cốt lõi để phát hiện ra vị trí.

Tuy nhiên, tín hiệu có thể bị sai đôi chút khi đi qua bầu khí quyển. Vì vậy, kèm theo thông điệp gửi tới các thiết bị nhận, các vệ tinh thường gửi kèm luôn thông tin về quỹ đạo và thời gian. Việc sử dụng đồng hồ nguyên tử sẽ đảm bảo chính xác về sự thống nhất thời gian giữa các thiết bị thu và phát.

Để biết vị trí chính xác của các vệ tinh, thiết bị nhận GPS còn nhận thêm 2 loại tín hiệu mã hóa:

Loại thứ nhất (được gọi là Almanac data) được cập nhật định kỳ và cho biết vị trí gần đúng của các vệ tinh trên quỹ đạo. Nó truyền đi liên tục và được lưu trữ trong bộ nhớ của thiết bị thu nhận khi các vệ tinh di chuyển quanh quỹ đạo.

Tuy nhiên, phần lớn các vệ tinh có thể hơi di chuyển ra khỏi quỹ đạo chính của chúng. Sự thay đổi này được ghi nhận bởi các trạm kiểm soát mặt đất. Việc sửa chữa những sai số này là rất quan trọng và được đảm nhiệm bởi trạm chủ trên mặt đất, trước khi thông báo lại cho các vệ tinh biết vị trí mới của chúng. Thông tin được sửa chữa này được gọi là Ephemeris data. Kết hợp Almanac data và Ephemeris data, các thiết bị nhận GPS biết chính xác vị trí của mỗi vệ tinh. Hiện nay, nếu có bản đồ điện tử, nhiều thiết bị nhận GPS sẽ hiển thị rõ ràng vị trí của bạn qua một màn hình, điều đó giúp cho việc định hướng trở nên cực kỳ thuận lợi. Nhưng nếu tắt thiết bị nhận tín hiệu trong khoảng thời gian chừng 5 giờ đồng hồ, nó sẽ mất đi các Almanac data (hay không còn nhận biết chính xác các vệ tinh trên quỹ đạo trái đất). Khi hoạt động trở lại, thiết bị sẽ cần khoảng thời gian chừng 30 giây để nạp lại thông tin về vị trí của vệ tinh, trước khi cho biết hiện thời bạn đang ở đâu.

### Các định dạng tín hiệu mà GPS Receiver nhận được

Để hiểu hơn về các tín hiệu mà các GPS Receiver nhận được từ đó có thể tính toán và hiển thị một cách thân thiện với người dùng, ở phần này chúng ta tìm hiểu về định dạng của các tín hiệu mà máy thu GPS nhận được.

Thông thường các máy thu GPS Receiver tiếp nhận thông tin từ các vệ tinh GPS theo định dạng:

* **NMEA 0183** (**NMEA):** là một kết hợp thiết bị điện và dữ liệu đặc điểm kỹ thuật cho các thông tin liên lạc giữa các thiết bị điện tử hàng hải như tiếng vọng âm vang, sonars (hệ thống định vị dưới mặt nước), anemometer (tốc độ và hướng gió), gyrocompass (La bàn), Autopilot (khối lái tự động), GPS Receiver (máy thu tín hiện GPS) và nhiều các loại công cụ khác. Nó đã được định nghĩa và được kiểm soát bởi **National Marine Electronics Association –** Hiệp hội Điện tử học Hàng hải quốc gia.
* **NMEA 0183** (**NMEA)**: là một chuẩn đơn giản sử dụng mã ASCII, được định nghĩa theo giao thức giao tiếp truyền tuần tự (Serial Communications Protocol) cho việc làm thế nào để có thể truyền phát một tín hiệu theo dạng “câu - sentence” từ một “người nói - talker” đến nhiều “người nghe - listeners” trong cùng một thời điểm. Thông qua một bộ mở rộng trung gian, một “talker” có thể có một cuộc “đàm thoại đơn hướng - unidirectional conversation” sử dụng các bộ dồn kênh (multiplexers), bộ nhận biết (sensors) thông qua một cổng đơn của máy tính đến số lượng “người nghe – listeners” không giới hạn ở gần đó.

Mỗi chuỗi/câu (sentence) của NMEA thường bắt đầu bằng ký hiệu $ (đây chính là quy ước của tầng giao thức ứng dụng trong NMEA – Applicatipon layer protocol Rules) và kết thúc bằng một “<CR><LF>” (carriage return linefeed). Các dữ liệu khác nhau được phân biệt bằng dấu “,” và đều phải được đưa vào đầy đủ trong một sentence như là một ký hiệu đánh dấu. Ví dụ: $GPGGA, , ,N, ,E, , , , ,M, ,M, ,0000\*4D.

Đối với một máy thu GPS, định dạng tín hiệu nhận được từ các vệ tinh GPS như sau:

$GPGGA,hhmmss.ss,ddmm.mmm,a,dddmm.mmm,b,q,xx,p.p,a.b,M,c.d,M,x.x,nnnn

|  |  |
| --- | --- |
| Ký hiệu | Chú thích |
| $ | Ký hiệu bắt đầu |
| GPGGA | Global Positioning System Fix Data – Khóa dữ liệu định vị toàn cầu. |
| hhmmss.ss | Thời gian theo định dạng UTC: thuộc kiểu Long. Ví dụ: 173948.823 |
| Ddmm.mmm | latitude of position – Vĩ độ của vị trí |
| A | N or S, latitutde hemisphere – Vĩ độ Nam hay Bắc |
| dddmm.mmm | Longtitude of position – Kinh độ của vị trí |
| B | E or W, longitude hemisphere – Kinh độ Đông hay Tây |
| Q | GPS Quality indicator (0=No fix, 1=Non-differential GPS fix, 2=Differential GPS fix, 6=Estimated fix) – Bộ phận chỉ chất lượng của GPS bao gồm 0: không xác định; 1=Không xác định vi sai GPS; 2 = Xác định vi sai GPS; 6 = ước lượng GPS. |
| Xx | Số vệ tinh đang sử dụng |
| p.p | Độ chính xác tín hiệu theo chiều ngang |
| a .b | Đo độ cao so với mặt nước biển |
| M | Các đơn vị đo độ cao |
| c.d | Độ cao của địa cầu (mặt đất) |
| M | Các đơn vị đo độ cao của địa cầu (mặt đất) |
| x.x | Thời gian vi sai của dự liệu |
| Nnnn | Xác định vi sai tham khảo tại trạm: 0000 – 1023 |

*Figure 25: Bảng chú thích về chuỗi tín hiệu của GPS Receiver*

## Độ chính xác của GPS và các nguồn lỗi của tín hiệu GPS

Các máy thu GPS ngày nay cực kì chính xác, nhờ vào thiết kế nhiều kênh hoạt động song song của chúng. Các máy thu 12 kênh song song (của Garmin) nhanh chóng khóa vào các quả vệ tinh khi mới bật lên và chúng duy trì chắc chắn liên hệ này, thậm chí trong tán lá rậm rạp hoặc thành phố với các toà nhà cao tầng. Tình trạng nhất định của khí quyển và các nguồn gây sai số khác có thể ảnh hưởng tới độ chính xác của máy thu GPS. Các máy thu GPS có độ chính xác trung bình trong vòng 15 mét.

Các máy thu mới hơn với khả năng WAAS (Hệ Tăng Vùng Rộng, *Wide Area Augmentation System*) có thể tăng độ chính xác trung bình tới dưới 3 mét. Không cần thêm thiết bị hay mất phí để có được lợi điểm của WAAS. Người dùng cũng có thể có độ chính xác tốt hơn với GPS Vi sai (*Differential GPS*, DGPS) sửa lỗi các tín hiệu GPS để có độ chính xác trong khoảng 3 đến 5 mét. Cục Phòng vệ Bờ biển Mỹ vận hành dịch vụ sửa lỗi này. Hệ thống bao gồm một mạng các đài thu tín hiệu GPS và phát tín hiệu đã sửa lỗi bằng các máy phát hiệu. Để thu được tín hiệu đã sửa lỗi, người dùng phải có máy thu tín hiệu vi sai bao gồm cả ăn-ten để dùng với máy thu GPS của họ.

Những điều có thể làm giảm tín hiệu GPS và vì thế ảnh hưởng tới độ chính xác bao gồm:

Giữ chậm của tầng đối lưu và tầng ion – Tín hiệu vệ tinh bị chậm đi khi xuyên qua tầng khí quyển.

Tín hiệu đi nhiều đường – Điều này xảy ra khi tín hiệu phản xạ từ nhà hay các đối tượng khác trước khi tới máy thu.

Lỗi đồng hồ máy thu – Đồng hồ có trong máy thu không chính xác như đồng hồ nguyên tử trên các vệ tinh GPS.

Lỗi quỹ đạo – Cũng được biết như lỗi thiên văn, do vệ tinh thông báo vị trí không chính xác.

Số lượng vệ tinh nhìn thấy – Càng nhiều quả vệ tinh được máy thu GPS nhìn thấy thì càng chính xác. Nhà cao tầng, địa hình, nhiễu loạn điện tử hoặc đôi khi thậm chí tán lá dầy có thể chặn thu nhận tín hiệu, gây lỗi định vị hoặc không định vị được. Nói chung máy thu GPS không làm việc trong nhà, dưới nước hoặc dưới đất.

Che khuất về hình học – Điều này liên quan tới vị trí tương đối của các vệ tinh ở thời điểm bất kì. Phân bố vệ tinh lí tưởng là khi các quả vệ tinh ở vị trí tạo các góc rộng với nhau. Phân bố xấu xảy ra khi các quả vệ tinh ở trên một đường thẳng hoặc cụm thành nhóm.

Sự giảm có chủ tâm tín hiệu vệ tinh – Là sự làm giảm tín hiệu cố ý do sự áp đặt của Bộ Quốc phòng Mỹ, nhằm chống lại việc đối thủ quân sự dùng tín hiệu GPS chính xác cao. Chính phủ Mỹ đã ngừng việc này từ tháng 5 năm 2000, làm tăng đáng kể độ chính xác của máy thu GPS dân sự. (Tuy nhiên biện pháp này hoàn toàn có thể được sử dụng lại trong những điều kiện cụ thể để đảm bảo gậy ông không đập lưng ông. Chính điều này là tiềm ẩn hạn chế an toàn cho dẫn đường và định vị dân sự).

# KẾT LUẬN

## Kết quả đạt được

### Ưu điểm

Thông qua quá trình nghiên cứu về lĩnh vực định vị toàn cầu (GPS) và phát triển hệ thống ***“Du Lịch Việt Nam”*** chúng tôi đã gặt hái được khá nhiều kinh nghiệm trong việc phát triển một ứng dụng chạy trên nên Web và di động sử dụng công nghệ cũng như là có những hiểu biết nhất định về hệ thống định vị toàn cầu.

Hiểu biết về cách thức hoạt động của hệ thống GPS, tín hiệu của các vệ tinh truyền dẫn như thế nào và quan trọng nhất là hiểu thêm về cách xác định vị trí của các vệ tinh GPS.

### Nhược điểm

Hệ thống chưa được phát triển nhưng ta có thể dự đoán được nhược điểm của chúng: ta cần xét đến một trong những hạn chế của việc phát triển là vấn đề xác thực (authentication) đối với việc phát triển ứng dụng cho các thiết bị di động. Trên thực tế, một ứng dụng có thể cài lên thiết bị di đông mà cụ thể là điện thoại di động phải được xác thực, nghĩa là nó đã được các tổ chức có thẩm quyền xác nhận về mặt an toàn và bảo mật đối với ứng dụng đó, các tổ chức đó có thể là nhà sản xuất các thiết bị di động hoặc là tổ chức thứ ba được ủy quyền bởi các nhà sản xuất thiết bị di động. Điều này cũng gây khó khăn không ít cho việc phát triển các ứng dụng trên thiết bị di động.

## Hướng phát triển

Giai đoạn ban đầu là hoàn thành các chức năng để có thể đáp ứng các yêu cầu đã đặt ra của đề tài.

Xây dựng thành công hệ thống “Du Lịch Việt Nam” trên nền Web và di động, đáp ứng được các chức năng của hệ thống:

* Dựa vào GPS, xác định vị trí của du khách
* Tìm đường, tìm thông tin cần thiết
* Xây dựng được webservices để ảnh chụp từ điện thoại được đăng lên website.
* Xây dựng được mạng xã hội du lịch trực tuyến

Chúng tôi rất muốn tiếp tục thực hiện đề tài này và sẽ làm kết quả cho luận văn sắp tới. Phía chúng mong muốn sẽ được thầy “Mai Văn Hà” cùng với anh Nguyễn Hữu Dũng phía bên công ty TNHH TT – KTS Toàn Cầu Xanh hỗ trợ thêm để hoàn thành đề tài này.

**DANH MỤC HÌNH**

[*Figure 1: Cấu trúc thư mục Cake* 6](#_Toc283670248)

[*Figure 2: The typical flow for PHP scripting* 7](#_Toc283670249)

[*Figure 3: How Cake makes use of the MVC structure* 8](#_Toc283670250)

[*Figure 4: Helpers* 11](#_Toc283670251)

[*Figure 5: Các tầng của CLDC J2ME* 18](#_Toc283670252)

[*Figure 6: Tiến trình xây dựng MIDlet* 20](#_Toc283670253)

[*Figure 7: J2ME và J2SE* 23](#_Toc283670254)

[*Figure 8: Bộ tiền kiểmt tra* 24](#_Toc283670255)

[*Figure 9: Mô hình sandbox* 24](#_Toc283670256)

[*Figure 10: Các mức tổ chức J2ME* 26](#_Toc283670257)

[*Figure 11: Triển khai hệ thống J2ME* 26](#_Toc283670258)

[*Figure 15: Bản đồ Google* 29](#_Toc283670259)

[*Figure 16: Minh họa các vệ tinh GPS và quỹ đạo của nó* 30](#_Toc283670260)

[*Figure 17: Minh họa sự hoạt động của hệ thống GPS* 32](#_Toc283670261)

[*Figure 18: Người A đang ở cách Hội An 30 Km* 33](#_Toc283670262)

[*Figure 19: Người A đang ở một trong hai điểm X và Y* 34](#_Toc283670263)

[*Figure 20: Xác định người A đang ở Đà Nẵng thông qua đường tròn thứ 3* 34](#_Toc283670264)

[*Figure 21: Khi X nhận tín hiệu từ một vệ tinh* 35](#_Toc283670265)

[*Figure 22: Khi X nhận tín hiệu từ 2 vệ tinh* 35](#_Toc283670266)

[*Figure 23: Điểm X khi nhận tín hiệu từ vệ tinh thứ 3* 36](#_Toc283670267)

[*Figure 24: Điểm X được xác định* 36](#_Toc283670268)

[*Figure 25: Bảng chú thích về chuỗi tín hiệu của GPS Receiver* 41](#_Toc283670269)

TÀI LIỆU THAM KHẢO

* 1. Trang web: <http://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_%C4%91%E1%BB%8Bnh_v%E1%BB%8B_to%C3%A0n_c%E1%BA%A7u>

<http://www.javavietnam.org/javavn/cms/vi/java_tutorials/java_tutorials_intermediate/J2ME_1.html>

<http://www.gps.gov/>

<http://www.nasm.si.edu/gps/work.html>

<http://scign.jpl.nasa.gov/learn/gps2.htm>

<http://www.trimble.com/gps/index.shtml>

<http://www.gps-home.com/howitworks.aspx>

<http://www.yssr.nb.ca/Documents/Files/Training/GPS_Training/gps%20overview_2008_09.ppt>

<http://www.maps-gps-info.com/gp.html>

<http://vi.wikipedia.org/wiki/Bluetooth>

<http://en.wikipedia.org/wiki/Global_Positioning_System>

<http://dientuvietnam.net/forums/archive/index.php/t-3059.html>

<http://en.wikipedia.org/wiki/NMEA>