TÀI LIỆU HƯỚNG DẪN XÂY DỰNG COMPONENT TRONG JOOMLA! 1.5

MỞ ĐẦU

Một component là một trong những thành phần mở rộng lớn nhất và phức tạp nhất. Các component có thể xem như các ứng dụng mini. Một cách trực quan chúng ta có thể hình dung joomla là một hệ điều hành và tất cả các component là các ứng dụng desktop. Mỗi trang (page) trong joomla sẽ gọi đến một component để tải về các nội dung chính (page body) của trang đó. Ví dụ component nội dung (com_content) là một ứng dụng mini xử lý tất cả các vấn đề về nội dung được trả lại, tương tự như thế component đăng ký là một ứng dụng mini nhằm xử lý việc đăng ký của người dùng. Một componet có hai phần chính: phần quản trị và phần trên site. Phần trên site là phần được sử dụng để tải về các trang khi được triệu gọi trong quá trình vận hành site thông thường. Phần quản trị cung cấp giao diện để cấu hình, quản lý các khía cạnh khác nhau của component và được truy cập thông qua ứng dụng quản trị của joomla.

Phần hướng dẫn dưới đây sẽ cung cấp cho các bạn một vài ví dụ hữu ích giúp bạn làm thế nào để tạo ra một component.

While we have gone to great lengths to make Joomla easy for content providers to use, we have equally spent a lot of time developing a flexible framework for developers to extend the capabilities of Joomla without having to touch the Core code.

PHẦN 1. PHÁT TRIỂN MỘT COMPONENT MVC.

(

1. Giới thiệu

Frameword mới trong joomla 1.5 đã mang lại những thuận lợi rất lớn cho các nhà phát triển. Các đoạn code đã hoàn toàn được kiểm tra và khá rõ ràng. Bài hướng dẫn này sẽ giúp bạn đi qua các bước trong quá trình phát triển nhằm tạo ra một component để có thể sử dụng frameword mới này.

Phạm vi của bài này là phát triển một component Hello World đơn giản. Trong các bài hướng dẫn tiếp theo, frameword đơn giản này sẽ được bổ sung, để thể hiện một cách đầy đủ khả năng và tính linh hoạt của mẫu thiết kế MVC trong joomla.

2. Yêu cầu

Bạn cần có joomla 1.5 hoặc các phiên bản mới hơn để thực hành bài hướng dẫn này.

3. Giới thiệu về MVC (Model-View-Controler)

Trong khi ý tưởng phía sau một component dường như là khá đơn giản thì, các đoạn code có thể nhanh chóng trở lên rất phức tạp khi các đặc điểm bổ sung được thêm vào hoặc giao diện được tùy biến.

Model-View-Controler (gọi tắt là MVC) là một mẫu thiết kế phần mềm được dùng để tổ chức các đoạn mã theo cách mà việc xử lý dữ liệu (business logic) và việc biểu diễn dữ liệu là tách rời nhau. Tiền đề nằm sau hướng tiếp cận này là nếu bussiness logic được nhóm vào trong một section thì giao diện và tương tác người dùng bao quanh dữ liệu có thể định dạng và tùy biến lại mà không có ảnh hưởng đến việc phải lập trình lại bussiness logic. (nghĩa là hình thức và nội dung là tách rời nhau, do đó khi thay đổi hình thức thể hiện thì không ảnh hưởng đến nội dung).

Có ba phần chính trong một MVC component (ba phần này bao gồm Model, View và Controler). Chúng được mô tả một cách vắn tắt ở dưới đây. Nếu bạn cần nhiều thông tin đầy đủ hơn xin vui lòng tham khảo thêm trong các đường dẫn được cung cấp ở cuối bài hướng dẫn này.

3.1. Model

Một model là thành phần của component đóng gói dữ liệu của ứng dụng. Nó thường cung cấp các thủ tục để quản lý và thao tác dữ liệu này theo một cách nào đó, trong đó có bổ sung thêm các thủ tục để lấy dữ liệu từ model. Trong trường hợp của chúng ta model sẽ chứa các phương thức như bổ sung, loại bỏ và cập nhật thông tin về những lời chào mừng trong cơ sở dữ liệu. Nó còn chứa phương thức để lấy danh sách các lời chào trong CSDL. Nói một cách tổng quát, việc truy cập vào CSDL lớp dưới sẽ được đóng gói trong model. Theo cách này, nếu một ứng dụng chuyển đổi sang việc sử dụng một file bình thường để lưu trữ thông tin của nó thay vì sử dụng CSDL, thì chỉ có thành phần model là thay đổi, các thành phần view và controler là không đổi.

3.2. View

View là một thành phần của component được sử dụng để trả lại dữ liệu từ model theo cách phù hợp với tương tác. Đối với các ứng dụng web, view thông thường là các trang HTML để trả lại dữ liệu. View lấy dữ liệu từ model (dữ liệu này được chuyển qua nó để tới controler). Và đưa dữ liệu vào trong template (dữ liệu sẽ hiển thị với người dùng). View không làm thay đổi dữ liệu. Nó chỉ hiển thị dữ liệu lấy từ model.

3.3. Controler

Controler chịu trách nhiệm phản hồi các hành động của người dùng. Trong các ứng dụng web, một hành động của người dùng thông thường là một yêu cầu tải trang. Controler sẽ xác định yêu cầu gì được đưa ra bởi người sử dụng và phản hồi thích hợp bằng việc yêu cầu model tính toán dữ liệu phù hợp và chuyển từ model vào view. Controler không thể hiện dữ liệu từ model, nó kích hoạt các phương thức trong model để hiệu chỉnh dữ liệu và sau đó chuyển từ model sang view để hiển thị dữ liệu.

4. Cài đặt Joomla! MVC

Trong joomla mẫu MVC được thực hiện sử dụng 3 lớp: <classname>JModel</classname>, <classname>JView</classname> và <classname>JController</classname>. Thông tin chi tiết về các lớp này vui lòng xem trong tài liệu hướng dẫn API ((WIP).

5. Tạo một component

Đối với component cơ sở của chúng ta, chúng ta chỉ cần 5 file:

- hello.php - đây là điểm vào cho component của chúng ta (this is the entry point to our component)
- controller.php – file này có chứa controler cơ bản (this file contains our base controller)
- views/hello/view.html.php – file này nhận các dữ liệu cần thiết và đặt nó lên template (this file retrieves the necessary data and pushes it into the template)
- views/hello/tmpl/default.php – file này là temple cho đầu ra (this is the template for our output)
- hello.xml – đây là một file XML nói cho joomla biết cách cài đặt component của chúng ta như thế nào (this is an XML file that tells Joomla! how to install our component).

5.1. Tạo entry point

Joomla luôn luôn được truy cập thông qua một điểm vào đơn: index.php cho các ứng dụng site và administrator/index.php cho ứng dụng quản trị. Sau đó ứng dụng sẽ tải các component cần thiết dựa trên giá trị chọn lựa trong URL hoặc trong dữ liệu POST. Đối với component của chúng ta URL sẽ như sau: index.php?option=com_hello&view=hello. Việc này sẽ tải file chính của chúng ta và có thể được xem như một điểm vào đơn cho component của chúng ta: components/com_hello/hello.php. Đoạn mã này thực sự là cụ thể tùy theo các component.

```
/**
 * @package    Joomla.Tutorials
```

```
 * @subpackage Components
 * components/com_hello/hello.php
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:modules/
 * @license    GNU/GPL
*/
// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
// Require the base controller
require_once( JPATH_COMPONENT.DS.'controller.php' );
// Require specific controller if requested
if($controller = JRequest::getWord('controller')) {
    $path = JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php';
    if (file_exists($path)) {
        require_once $path;
    } else {
        $controller = '';
    }
}
// Create the controller
$classname    = 'HelloController'.$controller;
$controller   = new $classname( );
// Perform the Request task
$controller->execute( JRequest::getVar( 'task' ) );
// Redirect if set by the controller
$controller->redirect();
?>
```

Câu lệnh đầu tiên là câu lệnh kiểm tra bảo mật.

JPATH_COMPONENT là đường dẫn tuyệt đối tới component hiện tại, trong trường hợp của chúng ta là components/com_hello. Nếu bạn cần xác định site component hoặc admin component thì bạn có thể sử dụng JPATH_COMPONENT và JPATH_COMPONENT_ADMINISTRATOR.

DS là dấu phân cách thư mục trong hệ thống của bạn: có thể là "\" hoặc "/". Điều này được thiết lập tự động bởi frameword, vì thế developer không phải quan tâm đến việc phát triển các phiên bản khác nhau cho các hệ điều hành khác nhau. DS sẽ luôn được sử dụng khi tham chiếu đến các file trên máy chủ cục bộ.

Sau khi tải controler cơ bản, chúng ta sẽ kiểm tra một controler cụ thể cần đến. Trong component này, controler cơ bản chỉ là một controler nhưng chúng ta đề cập đến điều này cho các công việc trong tương lai.

<classname>JRequest</classname>:getVar() tìm một biến trong URL hoặc POST dữ liệu. Bởi vậy nếu URL của chúng ta là:

index.php?option=com_hello>controller=controller_name

thì chúng ta có thể nhận được tên controler của chúng ta trong component bằng sử dụng câu lệnh sau: echo <classname>JRequest</classname>::getVar('controller');

Bây giờ chúng ta đã có controler cơ sở '<classname>HelloController</classname>' trong com_hello/controller.php, và nếu cần thiết bổ sung thêm các controler dạng như: '<classname>HelloControllerController1</classname>' trong com_hello/controllers/controller1.php. Sự sắp xếp theo hệ thống này sẽ tạo ra thuận lợi về sau: '{Componentname}{Controller}{Controllername}'.

Sau khi controler được tạo ra, chúng ta cho controler chạy nhiệm vụ như được chỉ ra trong URL: index.php?option=com_hello&task=sometask. Nếu không có nhiệm vụ nào được thiết lập thì nhiệm vụ mặc định 'display' sẽ được giả định. Khi 'display' được sử dụng, biến 'view' sẽ quyết định cái gì sẽ được hiển thị. Các nhiệm vụ khác như 'save', 'edit', 'new', …

Controler có thể quyết định redirect the page (thực hiện tải lại một trang), thông thường là sau khi một nhiệm vụ như 'save' được hoàn thành. Câu lệnh cuối cùng thực hiện việc này.

Điểm vào chính (hello.php) về bản chất đã thông qua việc điều khiển controler thực hiện các nhiệm vụ được đặt ra trong request.

5.2. Tạo controler

Component của chúng ta chỉ có một nhiệm vụ - greet the world (thể hiện lời chào). Bởi vậy controler sẽ rất đơn giản. Không cần đến việc tính toán trên dữ liệu. Tất cả những gì cần thiết phải làm là tải view thích hợp. Chúng ta sẽ chỉ có một phương thức trong controler là display(). Hầu hết các hàm cần thiết được xây dựng trong lớp JControler, bởi vậy tất cả những gì chúng ta cần là gọi phương thức JControler::display(); Code của controler cơ sở như sau:

```
<?php
/**
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:modules/
 * @license    GNU/GPL
 */
// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport('joomla.application.component.controller');
/**
```

```
 * Hello World Component Controller
 *
 * @package    Joomla.Tutorials
```

```
 * @subpackage Components
```

```
 */
class HelloController extends JController
{
    /**
     * Method to display the view
     *
     * @access    public
     */
    function display()
    {
        parent::display();
    }
}
?>
```

Việc khởi tạo của <classname>JController</classname> sẽ luôn luôn đăng ký một nhiệm vụ display() khi không có nhiệm vụ cụ thể nào được chỉ ra (bằng việc sử dụng phương thức registerDefaultTask()), nó sẽ được thiết lập như một nhiệm vụ mặc định.

Phương thức display() này thật sự không cần thiết bởi vì tất cả những gì nó làm là gọi hàm khởi tạo của cha nó. Tuy nhiên, đó là một cơ sở khá tốt để chỉ ra điều gì cần phải làm trong controler.

Phương thức <classname>JController</classname>::display() sẽ xác định view và layout từ request, tải view đó và thiết lập layout. Khi bạn tạo ra một menu item cho component của bạn, menu manager sẽ cho phép admin lựa chọn view mà họ thích để thể hiện và trên layout cụ thể. Một view thông thường được xem như một cách hiển thị của một tập các dữ liệu nào đó (ví dụ, danh sách các car, danh sách các event, một car đơn, một event đơn, …). Một layout là một cách tổ chức view.

Trong component của chúng ta chúng ta sẽ có một view đơn được gọi là hello, và một layout đơn (default).

5.3. Tạo view

Nhiệm vụ của view là rất đơn giản: nó nhận dữ liệu được thể hiện và đặt nó lên template. Dữ liệu được đặt lên template sử dụng phương thức <classname>JView</classname>::assignRef. Đoạn mã của view như sau:

```
<?php
/**
```

```
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:modules/
```

```
 * @license    GNU/GPL
```

```
*/
// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.view');
/**
 * HTML View class for the HelloWorld Component
 *
 * @package    HelloWorld
 */
class HelloViewHello extends JView
{
    function display($tpl = null)
    {
        $greeting = "Hello World!";
        $this->assignRef( 'greeting', $greeting );

        parent::display($tpl);
    }
}
?>
```

5.4. Tạo template

Joomla template/ layout là các file PHP thông thường, được sử dụng để bố trí, xếp đặt dữ liệu từ view theo một cách cụ thể nào đó. Các biến được gán bởi phương thức JView::assignRef có thể được truy cập từ template sử dụng $this→{propertyname} (xem mã template bên dưới như một ví dụ).

Template của chúng ta rất đơn giản, chúng ta chỉ muốn thể hiện một lời chào hợp quy cách từ view.

```
<?php // no direct access
defined('_JEXEC') or die('Restricted access'); ?>
<h1><?php echo $this->greeting; ?></h1>
```

5.5. Đóng gói tất cả - Tạo ra file hello.xml

Có thể cài đặt thủ công một component bằng cách copy tất cả các file bằng FPT client và hiệu chỉnh CSDL. Nhưng sẽ hiệu quả hơn nếu tạo ra một file được đóng gói để Joomla! Installer thực hiện điều này cho bạn. File đóng gói này sẽ chứa nhiều dạng thông tin khác nhau như:

- Các miêu tả chi tiết cơ bản về component của bạn (ví dụ như tên), và tùy ý một số mô tả thông tin về bản quyền ….
- Một danh sách các file cần copy.
- Một file PHP thực hiện bổ sung các thao tác cài đặt và gỡ bỏ (file này là không bắt buộc).
- Một file SQL có chứa các câu truy vấn dữ liệu mà sẽ được thực hiện vào lúc cài đặt hoặc gỡ bỏ (file này là không bắt buộc).

Định dạng của file XML như dưới đây:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">
<install type="component" version="1.5.0">
 <name>Hello</name>
 <!-- The following elements are optional and free of formatting conttraints -->
 <creationDate>2007 02 22</creationDate>
 <author>John Doe</author>
 <authorEmail>john.doe@example.org</authorEmail>
 <authorUrl>http://www.example.org</authorUrl>
 <copyright>Copyright Info</copyright>
 <license>License Info</license>
 <!-- The version string is recorded in the components table -->
 <version>Component Version String</version>
 <!-- The description is optional and defaults to the name -->
 <description>Description of the component ...</description>

 <!-- Site Main File Copy Section -->
 <!-- Note the folder attribute: This attribute describes the folder
      to copy FROM in the package to install therefore files copied
      in this section are copied from /site/ in the package -->
 <files folder="site">
  <filename>index.html</filename>
  <filename>hello.php</filename>
  <filename>controller.php</filename>
  <filename>views/index.html</filename>
  <filename>views/hello/index.html</filename>
```

```
  <filename>views/hello/view.html.php</filename>
  <filename>views/hello/tmpl/index.html</filename>
  <filename>views/hello/tmpl/default.php</filename>
 </files>
```

```
 <administration>
  <!-- Administration Menu Section -->
  <menu>Hello World!</menu>

  <!-- Administration Main File Copy Section -->
  <files folder="admin">
   <filename>index.html</filename>
   <filename>admin.hello.php</filename>
  </files>
 </administration>
</install>
```

Nếu bạn đã xem xét file này một cách cẩn thận bạn sẽ nhận thấy rằng có một vài file sẽ được copy mà chúng ta chưa đề cập ở trên. Chúng là các file index.html. Một file index.html được đặt trong mỗi thư mục để ngăn cản những người dùng tò mò trong việc liệt kê một danh sách thư mục. Nếu không có file index.html một vài máy chủ web sẽ liệt kê danh sách nội dung của thư mục. Điều này thường gây rắc rối. Các file này có một dòng đơn như sau: (Nó đơn giản là thể hiện một trang trống)

```
<html><body bgcolor="#FFFFFF"></body></html>
```

File khác là admin.hello.php, đây là điểm vào cho section quản trị trong component của chúng ta. Bởi vì chúng ta chưa có section quản trị trong component vào thời điểm này nên nó sẽ có nội dung tương tự như file index.html.

6. Những người viết và địa chỉ download ví dụ

Những người tham gia viết bài hướng dẫn này:

- mjaz
- staalanden

Component có thể được download tại:
http://dev.joomla.org/components/com_jd-wiki/data/media/components/com_hello1.zip

PHẦN 2. BỔ SUNG MODEL VÀO COMPONENT MVC.

(Phần hướng dẫn này được dịch từ trang web:
http://dev.joomla.org/component/option,com_jd-wiki/Itemid,/id,components:hello_world_mvc2/).

1. Giới thiệu

Trong bài hướng dẫn đầu tiên (ở phần 1), chúng ta đã mô tả việc tạo thành một component view-controler đơn giản sử dụng Joomla! Frameword.

Trong bài đầu tiên, lời chào mừng đã được code cố định vào trong view. Điều này không tuân theo mẫu MVC một cách chính xác bởi vì view chỉ được hiển thị dữ liệu chứ không chứa nó.

Trong phần thứ hai này, chúng ta sẽ mô tả làm thế nào để chuyển dữ liệu ra khỏi view và đưa nó vào một model. Trong các bài hướng dẫn tiếp theo chúng ta sẽ mô tả khả năng và sự mềm dẻo mà mẫu thiết kế cung cấp.

2. Tạo ra một model

Khái niệm model được gọi tên như thế bởi vì lớp này được mong đợi sẽ mô hình hóa cho một vài thực thể nào đó. Trong trường hợp của chúng ta, model đầu tiên sẽ đưa ra một lời chào mừng. Điều này phù hợp với thiết kế hiện tại, bởi vì chúng ta đã có một view 'hello', view đó thể hiện một câu chào mừng tới người dùng.

Cách thông thường để đặt tên cho các model trong Joomla! Frameword là tên lớp bắt đầu bằng tên của component (trong trường hợp của chúng ta là 'hello'), tiếp theo là 'model', và cuối cùng là là tên của model. Bởi thế lớp model của chúng ta được gọi là HelloModelHello.

Ở thời điểm này chúng ta chỉ mô hình hóa cách xử lý của model hello, và nó sẽ trả lại một lời chào. Chúng ta sẽ có một phương thức được gọi là getGreeting(). Phương thức này đơn giản là trả lại chuỗi "Hello, World". Dưới đây là code cho model của chúng ta:

```
<?php
/**
 * Hello Model for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:modules/
 * @license    GNU/GPL
 */
// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die();
jimport( 'joomla.application.component.model' );
/**
 * Hello Model
```

```
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HelloModelHello extends JModel
{
    /**
    * Gets the greeting
    * @return string The greeting to be displayed to the user
    */
    function getGreeting()
    {
        return 'Hello, World!';
    }
}
```

Trong đoạn code trên, chú ý dòng bắt đầu là lệnh jimport. Hàm jimport được sử dụng để tải các file từ Joomla! frameword cần thiết cho component của chúng ta. Cụ thể câu lệnh trên sẽ tải file /libraries/joomla/application/component/model.php. Các dấu chấm '.' được sử dụng như các dấu phân cách thư mục, và phần cuối cùng là tên file cần tải. Tất cả các file được tải từ thư mục libraries. File trong trường hợp trên có chứa định nghĩa của lớp JModel, điều này là cần thiết vì lớp của chúng ta được kế thừa từ lớp này.

Bây giờ chúng ta đã tạo ra model của mình. Tiếp theo chúng ta sẽ phải hiệu chính view để nó có thể lấy được lời chào.

3. Sử dụng model

Joomla! Frameword được thiết lập theo cách controler sẽ tự động tải model có tên trùng với view và đặt nó vào trong view. Vì view của chúng ta được đặt là 'hello', nên model 'hello' của chúng ta sẽ tự động được tải và được đặt vào trong view. Bởi vậy chúng ta sẽ dễ dàng nhận được một tham chiếu đến model của chúng ta sử dụng phương thức JView::getModel(). Đoạn mã cho view trước của chúng ta có chứa dòng:

$greeting = "Hello World!";

Để sử dụng model, chúng ta chuyển đổi dòng trên thành:

```
$model =& $this->getModel();
$greeting = $model->getGreeting();
```

Bây giờ, code của view sẽ như sau:

```
<?php
/**
 * Hello View for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:modules/
 * @license    GNU/GPL
 */
// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.view');
/**
 * HTML View class for the HelloWorld Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HelloViewHello extends JView
{
    function display($tpl = null)
    {
        $model =& $this->getModel();
        $greeting = $model->getGreeting();
        $this->assignRef( 'greeting', $greeting );

        parent::display($tpl);
    }
}
?>
```

4. Bổ sung file vào package

Tất cả những gì còn lại cần phải làm là bổ sung một entry vào file XML để model mới của chúng ta được copy. Joomla! Frameword sẽ tìm kiếm model của chúng ta trong thư mục models. Bởi vậy entry cho file này trông như sau:

```
<filename>models/hello.php</filename>
```

File hello.xml mới của chúng ta sẽ như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">
<install type="component" version="1.5.0">
        <name>Hello</name>
    <!-- The following elements are optional and free of formatting conttraints -->
    <creationDate>2007 02 22</creationDate>
    <author>John Doe</author>
    <authorEmail>john.doe@example.org</authorEmail>
    <authorUrl>http://www.example.org</authorUrl>
    <copyright>Copyright Info</copyright>
    <license>License Info</license>
    <!--  The version string is recorded in the components table -->
    <version>Component Version String</version>
    <!-- The description is optional and defaults to the name -->
    <description>Description of the component ...</description>

    <!-- Site Main File Copy Section -->
    <files folder="site">
        <filename>index.html</filename>
        <filename>hello.php</filename>
        <filename>controller.php</filename>
        <filename>views/index.html</filename>
        <filename>views/hello/index.html</filename>
        <filename>views/hello/view.html.php</filename>
        <filename>views/hello/tmpl/index.html</filename>
        <filename>views/hello/tmpl/default.php</filename>
        <filename>models/index.html</filename>
        <filename>models/hello.php</filename>
    </files>

    <administration>
        <!-- Administration Menu Section -->
        <menu>Hello World!</menu>

        <!-- Administration Main File Copy Section -->
        <!-- Note the folder attribute: This attribute describes the folder
```

```
            to copy FROM in the package to install therefore files
copied
            in this section are copied from /admin/ in the package -->
        <files folder="admin">
    <!-- Site Main File Copy Section -->
            <filename>index.html</filename>
            <filename>admin.hello.php</filename>
        </files>
    </administration>
</install>
```

5. Kết luận

Chúng ta đã có một component MVC đơn giản. Mỗi phần tử là rất đơn giản vào thời điểm này, nhưng nó cung cấp một khả năng và sự linh hoạt rất lớn.

6. Những người viết và địa chỉ download ví dụ

Những người tham gia viết bài hướng dẫn này:

- staalanden

Component có thể được download tại:
http://dev.joomla.org/components/com_jd-wiki/data/media/components/com_hello2.zip

PHẦN 3. SỬ DỤNG CSDL TRONG COMPONENT MVC.

(Phần hướng dẫn này được dịch từ trang web:
http://dev.joomla.org/component/option,com_jd-wiki/Itemid,/id,components:hello_world_mvc3/).

1. Giới thiệu

Trong hai bài hướng dẫn đầu tiên (phần 1 và phần 2) chúng tôi đã hướng dẫn các bạn làm thế nào để xây dựng một component MVC đơn giản. Chúng ta đã có một view lấy dữ liệu từ model (trong bài hướng dẫn thứ 2). Trong bài này chúng ta sẽ làm việc với model. Thay vì việc dữ liệu được code trực tiếp trong model, thì model sẽ lấy dữ liệu từ một bảng trong CSDL.

2. Lấy dữ liệu

Model hiện tại của chúng ta có một phương thức getGreeting(). Phương thức này rất đơn giản, tất cả những gì nó làm là trả về một câu chào được code cố định.

Để làm những điều này thú vị hơn, chúng ta sẽ lấy câu chào từ bảng cơ sở dữ liệu. Trong phần sau, chúng ta sẽ mô tả làm cách nào để tạo ra một file SQL và bổ sung các đoạn mã phù hợp vào file XML để tạo ra bảng và bổ sung dữ liệu mẫu vào bảng khi component được cài đặt. Bây giờ, chúng ta chỉ đơn giản thay thế câu lệnh trả về trong hàm getGreeting bằng đoạn code lấy lời chào mừng từ CSDL và trả lại lời chào mừng đó.

Đầu tiên, chúng ta cần lấy được một tham chiếu đến đối tượng CSDL. Vì joomla sử dụng CSDL cho các thao tác thông thường của nó, nên nó đã có một kết nối tới CSDL tồn tại, bởi thế không cần thiết phải tạo ra một kết nối CSDL riêng. Có thể lấy được một tham chiếu tới kết nối bằng cách sử dụng câu lệnh:

```
$db =& JFactory::getDBO();
```

JFactory là một lớp static được sử dụng để nhận tham chiếu đến nhiều đối tượng hệ thống khác nhau. Các thông tin thêm về lớp này có thể tham khảo trong tài liệu về API.

Tên hàm getDBO là viết tắt cho get DataBase Object, có thể nhớ một cách dễ dàng và rất quan trọng.

Bây giờ chúng ta đã có một tham chiếu đến đối tượng CSDL, Chúng ta có thể lấy dữ liệu. Điều này được thực qua hai bước:

- Lưu trữ câu truy vấn của vào đối tượng CSDL
- Tải kết quả về

Code của phương thức getGreeting() bây giờ sẽ như sau:

```
function getGreeting()
{
   $db =& JFactory::getDBO();

   $query = 'SELECT greeting FROM #__hello';
   $db->setQuery( $query );
```

```
    $greeting = $db->loadResult();

    return $greeting;
}
```

Trong đoạn mã trên 'hello' là tên của bảng trong CSDL mà chúng ta sẽ tạo ra trong phần sau. Và 'greeting' là tên của trường lưu trữ câu chào mừng. Nếu bạn chưa quen với SQL, sẽ rất hữu ích nếu bạn lấy một bài hướng dẫn để xem xét lại. Những bài hướng dẫn như thế bạn có thể tìm thấy ở w3schools.

Phương thức $db→loadResult() sẽ chạy câu lệnh truy vấn CSDL đã được lưu trữ và trả về trường đầu tiên của dòng đầu tiên trong kết quả. Xem JDatabase API reference để có các thông tin nhiều hơn về các phương thức khác trong lớp JDatabase.

3. Tạo ra file SQL trong quá trình cài đặt và gỡ bỏ component

Joomla! installer có xây dựng sẵn việc hỗ trợ đối với việc chạy các câu truy vấn trong quá trình cài đặt component. Tất cả các câu truy vấn này được lưu trữ trong một file text chuẩn.

Chúng ta sẽ có 3 câu truy vấn trong file cài đặt: việc đầu tiên sẽ thực hiện xóa bảng trong trường hợp nó đã tồn tại, tiếp theo thực hiện việc tạo ra bảng với các trường phù hợp, và cuối cùng là chèn dữ liệu vào bảng vừa tạo.

Dưới đây là các câu truy vấn của chúng ta:

```
DROP TABLE IF EXISTS `#__hello`;

CREATE TABLE `#__hello` (
  `id` int(11) NOT NULL auto_increment,
  `greeting` varchar(25) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT DEFAULT CHARSET=utf8;

INSERT INTO `#__hello` (`greeting`) VALUES ('Hello, World!'),
('Bonjour, Monde!'),
('Ciao, Mondo!');
```

Bạn có thể thấy tiền tố trong các tên bảng hởi kỳ cục. Tuy nhiên, Joomla! sẽ thay thế các tiền tố này bởi tiền tố đã được xác định trong quá trình cài đặt. Đối với hầu hết các bản cài đặt, bảng này sẽ trở thành jos_hello. Điều này cho phép nhiều bản cài đặt joomla có thể sử dụng chung CSDL, và ngăn chặn các xung đột với các ứng dụng khác sử dụng cùng tên bảng (ví dụ, hai ứng dụng có thể chia sẻ chung một CSDL, nhưng cả hai đều cần một bảng 'user', với cách này vấn đề sẽ được giải quyết).

Chúng ta đã xác định hai trường trong CSDL. Trường đầu tiên là 'id' đóng vai trò là khóa chính. Khóa chính trong một bảng của CSDL là trường dùng để xác định duy nhất

một bản ghi trong bảng. Nó thường được sử dụng để tìm kiếm các hàng trong CSDL. Trường còn lại là 'greeting'. Đây là trường lưu câu chào mừng được trả lại bởi truy vấn mà chúng ta đã sử dụng ở trên. Các câu truy vấn của chúng ta sẽ được lưu trong file install.utf.sql.

Tạo ra file SQL để gỡ bỏ cài đặt

Mặc dù chúng ta luôn mong muốn người dùng sẽ không bao giờ muốn gỡ bỏ component của chúng ta, tuy nhiên có một điều quan trọng là nếu họ thực hiện việc đó chúng ta không được để lại bất cứ thứ gì sau khi gỡ bỏ. Joomla! sẽ quản lý việc xóa các file và các thư mục được tạo ra trong quá trình cài đặt, nhưng bạn phải bổ sung thủ công các câu truy vấn sẽ loại bỏ các bảng đã được thêm vào CSDL. Trong ví dụ, chúng ta chỉ tạo một bảng, nên chúng ta chỉ cần một câu truy vấn để xóa nó.

```
DROP TABLE IF EXISTS `#__hello`;
```

Câu truy vấn này sẽ được lưu trữ trong file uninstall.utf.sql.

4. Cập nhật file cài đặt

Chúng ta cần phải thay đổi một vài thứ trong file cài đặt XML. Đầu tiên, chúng ta cần bổ sung hai file mới vào danh sách các file. File cài đặt SQL sẽ phải đưa vào trong thư mục admin. Thứ hai, chúng ta cần phải báo cho trình cài đặt chạy các câu truy vấn trong các file của chúng ta trong quá trình cài đặt và quá trình gỡ bỏ.

File XML mới của chúng ta sẽ như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE install SYSTEM "http://dev.joomla.org/xml/1.5/component-install.dtd">
<install type="component" version="1.5.0">
   <name>Hello</name>
   <!-- The following elements are optional and free of formatting conttraints -->
   <creationDate>2007 02 22</creationDate>
   <author>John Doe</author>
   <authorEmail>john.doe@example.org</authorEmail>
   <authorUrl>http://www.example.org</authorUrl>
   <copyright>Copyright Info</copyright>
   <license>License Info</license>
   <!-- The version string is recorded in the components table -->
   <version>Component Version String</version>
   <!-- The description is optional and defaults to the name -->
   <description>Description of the component ...</description>
```

```
   <!-- Site Main File Copy Section -->
```

```
   <files folder="site">
      <filename>index.html</filename>
      <filename>hello.php</filename>
      <filename>controller.php</filename>
      <filename>views/index.html</filename>
      <filename>views/hello/index.html</filename>
      <filename>views/hello/view.html.php</filename>
      <filename>views/hello/tmpl/index.html</filename>
      <filename>views/hello/tmpl/default.php</filename>
      <filename>models/hello.php</filename>
   </files>
   <install>
      <sql>
         <file charset="utf8" driver="mysql">install.sql</file>
      </sql>
   </install>
   <uninstall>
      <sql>
         <file charset="utf8" driver="mysql">uninstall.sql</file>
      </sql>
   </uninstall>
   <administration>
      <!-- Administration Menu Section -->
      <menu>Hello World!</menu>

      <!-- Administration Main File Copy Section -->
      <!-- Note the folder attribute: This attribute describes the
folder
         to copy FROM in the package to install therefore files copied
         in this section are copied from /admin/ in the package -->
      <files folder="admin">
   <!-- Site Main File Copy Section -->
         <filename>index.html</filename>
         <filename>admin.hello.php</filename>
         <filename>install.sql</filename>
         <filename>uninstall.sql</filename>
      </files>
   </administration>
```

```
</install>
```

Chú ý đến hai thuộc tính xuất hiện trong các thẻ <file> nằm trong các phần <install> và <uninstall>: charset và driver. Charset là kiểu charset được sử dụng. Chỉ có charset hợp lệ là utf8. Nếu bạn muốn tạo ra các file cài đặt cho CSDL không phải là utf8 bạn sẽ bỏ qua thuộc tính này. Thuộc tính driver xác định CSDL nào các câu truy vấn được viết cho. Hiện tại, thuộc tính này chỉ có thể là mysql, nhưng các phiên bản joomla trong tương lai có thể có nhiều dạng CSDL hơn.

5. Kết luận

Cho đến bây giờ chúng ta có một component tận dụng được các thuận lợi của cả các lớp Joomla! MVC framework và lớp JDatabase. Bây giờ bạn đã có khả năng viết các component MVC tương tác với CSDL và có thể sử dụng Joomla! Installer để tạo và thao tác trên các bảng CSDL.

6. Những người viết và địa chỉ download ví dụ

Những người tham gia viết bài hướng dẫn này:

- staalanden

Component có thể được download tại:
http://dev.joomla.org/components/com_jd-wiki/data/media/components/com_hello3.zip

PHẦN 4. TẠO GIAO DIỆN QUẢN TRỊ TRONG MVC.

(Phần hướng dẫn này được dịch từ trang web:
http://dev.joomla.org/component/option,com_jd-wiki/Itemid,/id,components:hello_world_mvc4/).

1. Giới thiệu

Trong ba bài hướng dẫn trước (phần 1, 2 và 3) chúng ta đã phát triển một component MVC lấy dữ liệu từ một bảng trong CSDL. Cho đến thời điểm hiện tại component của chúng ta chưa có cách nào để bổ sung thêm dữ liệu tới CSDL, ngoại trừ việc thực hiện điều này một cách thủ công bằng một công cụ khác. Trong phần này chúng ta sẽ phát triển một section quản trị cho component, để có khả năng quản lý các entry trong CSDL.

2. Tạo ra frameword cơ sở

Frameword cơ sở của panel quản trị cũng tương tự như phần site. Điểm vào chính cho phần quản trị là file admin.hello.php. File này giống y hệt file hello.php được sử dụng trong phần site ngoại trừ tên của controler mà nó tải sẽ được thay đổi thành HellosController. Ngoài ra, controler mặc định được gọi là controller.php và file này giống y hệt controler mặc định trong phần site, ngoại trừ tên của controler được gọi là HellosController thay vì HelloController. Sự thay đổi này để JControler theo mặc định sẽ tải hellos view, view này sẽ hiển thị một danh sách các câu chào mừng.

Dưới đây là code của admin.hello.php

```
<?php
/**
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:components/
 * @license    GNU/GPL
*/
// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
// Require the base controller
require_once( JPATH_COMPONENT.DS.'controller.php' );
// Require specific controller if requested
if($controller = JRequest::getWord('controller')) {
    $path = JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php';
    if (file_exists($path)) {
        require_once $path;
    } else {
        $controller = '';
    }
```

```
}
// Create the controller
```

```
$classname    = 'HellosController'.$controller;
$controller   = new $classname( );
// Perform the Request task
$controller->execute( JRequest::getVar( 'task' ) );
// Redirect if set by the controller
$controller->redirect();
?>
```

Chúng sẽ bắt đầu với view hellos và model hellos. Đầu tiên là model hellos.

2.1. Model Hellos

Model Hellos sẽ rất đơn giản. Hiện tại, công việc mà chúng ta cần chỉ là truy vấn để nhận được một danh sách của các câu chào mừng từ CSDL. Công việc này sẽ được thực hiện bằng cách gọi phương thức getData().

Lớp JModel có một phương thức protected được xây dựng sẵn _getList(). Phương thức này có thể được sử dụng để đơn giản hóa tác vụ truy vấn để nhận được một danh sách các bản ghi từ CSDL. Đơn giản là chúng ta cần truyền cho nó câu truy vấn và nó sẽ trả lại danh sách các bản ghi.

Trong tương lai, có thể chúng ta lại muốn sử dụng câu truy vấn trong một phương thức khác. Vì vậy, ở đây chúng ta sẽ tạo ra một phương thức private _buildQuery() sẽ trả lại câu truy vấn sẽ được truyền vào phương thức _getList(). Điều này sẽ tạo ra khả năng thay đổi câu truy vấn khi nó được đặt trong một tình huống khác.

Do đó trong lớp của chúng ta sẽ có hai phương thức: getData() và _buildQuery().

Phương thức _buidQuery đơn giản là trả về câu truy vấn. Code của nó như sau:

```
/**
 * Returns the query
 * @return string The query to be used to retrieve the rows from the
database
 */
function _buildQuery()
{
    $query = ' SELECT * '
           . ' FROM #__hello '
    ;
    return $query;
}
```

Phương thức getData() nhận câu truy vấn và trả lại danh sách các bản ghi từ CSDL. Bây giờ có thể xảy ra tình huống là chúng ta cần nhận được danh sách này 2 lần trong việc tải một trang. Thật là lãng phí khi truy vấn CSDL 2 lần. Bởi vậy chúng ta sẽ phải làm cho phương thức này lưu trữ dữ liệu trong một thuộc tính protected để trong các yêu cầu tiếp theo, nó chỉ đơn giản là trả về dữ liệu mà nó đã lấy được. Thuộc tính này sẽ được gọi là _data.

Dưới đây là code của phương thức getData()

```
/**
 * Retrieves the hello data
 * @return array Array of objects containing the data from the database
 */
function getData()
{
    // Lets load the data if it doesn't already exist
    if (empty( $this->_data ))
    {
        $query = $this->_buildQuery();
        $this->_data = $this->_getList( $query );
    }
    return $this->_data;
}
```

Code đầy đủ của model sau khi hoàn thành như sau:

```
<?php
/**
 * Hellos Model for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:components/
 * @license        GNU/GPL
 */
// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die();
jimport( 'joomla.application.component.model' );
/**
 * Hello Model
 *
```

```
 * @package    Joomla.Tutorials
 * @subpackage Components
```

```
 */
class HellosModelHellos extends JModel
{
    /**
     * Hellos data array
     *
     * @var array
     */
    var $_data;

    /**
     * Returns the query
     * @return string The query to be used to retrieve the rows from
the database
     */
    function _buildQuery()
    {
        $query = ' SELECT * '
            . ' FROM #__hello '
        ;
        return $query;
    }
    /**
     * Retrieves the hello data
     * @return array Array of objects containing the data from the
database
     */
    function getData()
    {
        // Lets load the data if it doesn't already exist
        if (empty( $this->_data ))
        {
            $query = $this->_buildQuery();
            $this->_data = $this->_getList( $query );
        }
        return $this->_data;
    }
```

```
}
```

File này được lưu lại thành models/hellos.php.

2.2. View hellos

Bây giờ chúng ta đã có một model để nhận dữ liệu, chúng ta cần hiển thị nó. View hellos này sẽ khá giống với view trong phần site. Model của chúng ta được tự động cài đặt trên site, bởi vậy nó nằm trong phần quản trị. Các phương thức bắt đầu bằng 'get' trong model có thể được truy cập bằng việc sử dụng phương thức get() của lớp JView. Bởi vậy view của chúng ta sẽ có 3 dòng. Một để lấy dữ liệu từ model, một để đẩy dữ liệu lên template, và một gọi phương thức display để hiển thị. Bởi vậy chúng ta có code như sau:

```
<?php
/**
 * Hellos View for Hello World Component
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 *          @link          http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:components/
 * @license        GNU/GPL
 */
// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die();
jimport( 'joomla.application.component.view' );
/**
 * Hellos View
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class HellosViewHellos extends JView
{
    /**
     * Hellos view display method
     * @return void
     **/
    function display($tpl = null)
    {
```

```
        JToolBarHelper::title(    JText::_(    'Hello    Manager'    ),
'generic.png' );
        JToolBarHelper::deleteList();
        JToolBarHelper::editListX();
```

```
        JToolBarHelper::addNewX();
```

```
        // Get data from the model
        $items =& $this->get( 'Data');
        $this->assignRef( 'items', $items );
        parent::display($tpl);
    }
}
```

File này được lưu thành views/hellos/view.html.php.

2.3. Template hellos

Template sẽ lấy dữ liệu được đưa đến từ view và đưa dữ liệu đó ra đầu ra để hiển thị. Chúng ta sẽ hiển thị kết quả ở đầu ra trong một bảng đơn giản. Trong khi template ở phần site là rất đơn giản, thì trong trang quản trị chúng ta sẽ cần mở rộng một chút để xử lý vòng lặp thông qua dữ liệu.

Dưới đây là code trong template của chúng ta:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>
<form action="index.php" method="post" name="adminForm">
<div id="editcell">
    <table class="adminlist">
    <thead>
        <tr>
            <th width="5">
                <?php echo JText::_( 'ID' ); ?>
            </th>
            <th>
                <?php echo JText::_( 'Greeting' ); ?>
            </th>
        </tr>
    </thead>
    <?php
    $k = 0;
    for ($i=0, $n=count( $this->items ); $i < $n; $i++)
    {
```

```
        $row =& $this->items[$i];
        ?>
        <tr class="<?php echo "row$k"; ?>">
            <td>
                <?php echo $row->id; ?>
```

```
            </td>
```

```
            <td>
                <?php echo $row->greeting; ?>
            </td>
        </tr>
        <?php
        $k = 1 - $k;
    }
    ?>
    </table>
</div>

<input type="hidden" name="option" value="com_hello" />
<input type="hidden" name="task" value="" />
<input type="hidden" name="boxchecked" value="0" />
<input type="hidden" name="controller" value="hello" />

</form>
```

Template này được lưu thành views/hellos/tmpl/default.php.

Chú ý rằng template của chúng ta được gói trong một form. Mặc dù điều này là không cần thiết vào thời điểm hiện tại, nhưng trong tương lai nó sẽ cần. Bây giờ chúng ta đã hoàn thànhm phần cơ bản của view đầu tiên. Chúng ta đã bổ sung thêm 5 file vào phần quản trị trong component:

- admin.hello.php
- controller.php
- models/hellos.php
- views/hellos/view.html.php
- views/hellos/tmpl/default.php

Bây giờ bạn có thể bổ sung các file này vào file cài đặt XML. Và thử lại.

3. Bổ sung thêm chức năng

Section quản trị của chúng ta cho đến thời điểm hiện tại chưa thực hiện nhiều tác vụ. Nó thực sự chưa làm điều gì. Những gì nó thực sự làm là hiển thị những entry mà chúng

ta có trong CSDL. Để làm nó hữu dụng hơn chúng ta cần bổ sung thêm các button và các đường link.

3.1. Toolbar

Chúng ta có thể thấy toolbar xuất hiện ở phía trên các panel khác trong component quản trị của joomla. Component của chúng ta cũng cần một thứ tương tự như thế. Chúng ta sẽ bổ sung thêm các nút xóa bản ghi, hiệu chỉnh bản ghi, thêm bản ghi mới. Ngoài ra chúng ta sẽ còn bổ sung thêm một title thể hiện trên toolbar.

Điều này được thực hiện bằng cách bổ sung thêm code vào view. Để bổ sung thêm các button chúng ta sử dụng các phương thức tĩnh từ lớp JToolbarHepper trong joomla. Code sẽ trông như sau:

```
JToolBarHelper::title(   JText::_( 'Hello Manager' ), 'generic.png' );
JToolBarHelper::deleteList();
JToolBarHelper::editListX();
JToolBarHelper::addNewX();
```

Ba phương thức này sẽ tạo ra các button phù hợp. Phương thức deleteList() có thể tùy chọn lấy ba tham số: Tham số đầu tiên là một chuỗi để hiển thị đến người dùng yêu cầu họ xác nhận việc xóa bản ghi. Tham số thứ hai là nhiệm vụ sẽ được gửi với câu truy vấn (mặc định là ‘remove’). Và tham số thứ 3 là đoạn text sẽ được thể hiện ở dưới button. Các phương thức editLisstX() và addNewX() có thể tùy chọn hai tham số: Tham số đầu tiên là nhiệm vụ (theo mặc định lần lượt là edit và add), và tham số thứ 2 là đoạn text sẽ được thể hiện ở dưới button.

Chú ý rằng cách sử dụng phương thức JText::_ trong template trước đây và ở đây là tương tự nhau. Đây là chức năng giúp cho component có thể chuyển đổi một cách rất dễ dàng. Phương thức JText::_ sẽ tìm kiếm xâu trong file ngôn ngữ của component và trả lại chuỗi đã được chuyển đổi. Nếu không có sự chuyển đổi nào được tìm thấy nó sẽ trả lại chính chuỗi đã truyền vào cho nó. Nếu chúng ta muốn chuyển đổi component sang một ngôn ngữ khác, tất cả những gì phải làm là tạo ra một file ngôn ngữ để ánh xạ xâu trong dấu trích dẫn với xâu đã được chuyển đổi tương ứng. [Ví dụ: nếu trong file ngôn ngữ chúng ta có một cặp ‘Greeting’ – ‘Hello world’ thì phương thức JText::_(‘Greeting’) sẽ trả lại chuỗi ‘Hello World’. Khi chúng ta cần chuyển qua tiếng Việt, trong file ngôn ngữ chúng ta chỉ cần có một cặp ‘Greeting’ – ‘Xin chào!’ lúc này phương thức JText::_(‘Greeting’) sẽ trả lại chuỗi ‘Xin chào!’.]

3.2. Checkbox và link

Bây giờ chúng ta đã có hai button. Hai button này thao tác trên các bản ghi đã tồn tại. Nhưng làm thế nào để chúng ta biết chúng ta đang thao tác trên bản ghi nào? Chúng ta phải yêu cầu người dùng nói cho chúng ta biết điều đó. Để làm điều này chúng ta cần bổ sung thêm các checkbox vào bảng hiển thị trên template để người dùng có thể lựa chọn các bản ghi thực sự.

Để bổ sung thêm các chekbox chúng ta cần bổ sung thêm một cột vào bảng. Chúng ta sẽ bổ sung cột này vào giữa hai cột đã có ở trên.

Trong phần đầu của cột, chúng ta sẽ thêm một checkbox để có thể lựa chọn tất cả các checkbox dưới nó thành on hoặc off.

```
<th width="20">
    <input type="checkbox" name="toggle" value=""
onclick="checkAll(<?php echo count( $this->items ); ?>);" />
```

```
</th>
```

Ở đây, hàm checkAll trong Javascript là một hàm được xây dựng trong Joomla! dựa trên gói Javascript sẽ cung cấp những chức năng mà chúng ta muốn.

Bây giờ chúng ta cần bổ sung các checkbox vào các từng hàng. Lớp JHTML của Joomla có môt phương thức JHTML::_() để tạo ra checkbox. Chúng ta sẽ bổ sung dòng code dưới đây vào vòng lặp.

```
$checked    = JHTML::_( 'grid.id', $i, $row->id );
```

Ngay phía sau dòng:

```
$row =& $this->items[$i];
```

Sau đó chúng ta sẽ thêm các cell của cột checkbox nằm giữa hai cột đã có (cột id và cột greeting):

```
<td>
    <?php echo $checked; ?>
</td>
```

Đến lúc này, để hiệu chỉnh một bản ghi, chúng ta cần phải check vào box tương ứng với bản ghi đó, sau đó click vào nút edit. Việc này có vẻ hơi rườm rà. Bởi thế chúng ta sẽ bổ sung một đường link đi trực tiếp đến form hiệu chỉnh câu chào mừng. Bổ sung thêm dòng dưới đây sau khi gọi phương thức JHTML::_() để tạo ra link HTML.

```
$link = JRoute::_(
'index.php?option=com_hello>controller=hello>task=edit>cid[]='. $row-
>id );
```

Sau đó đưa đường link vào trong cell chứa câu chào mừng:

```
<td>
    <a href="<?php echo $link; ?>"><?php echo $row->greeting; ?></a>
</td>
```

Chú ý rằng đường link này chỉ đến controler hello. Controler này sẽ xử lý việc thao tác trên dữ liệu. Nếu xem lại code trong phần template bạn sẽ thấy chúng ta sẽ có 4 trường hidden input ở phía dưới form. Trường đầu tiên có là 'option', trường này cần thiết để xác định chúng ta đang ở trong component com_hello. Trường thứ hai là nhiệm vụ cần thực hiện, trường này sẽ lấy các thiết lập khi một trong số các nút trên toolbar được click, Nếu trường này bị bỏ trống thì chúng ta sẽ báo lỗi và các button sẽ được disable. Trường hidden input thứ ba là trường boxchecked, trường này sẽ giữ số lượng box được check. Các button edit và delete sẽ kiểm tra để chắc chắn rằng số này là lớn hơn 0, và chúng sẽ không cho phép form được submit nếu điều kiện trên không thỏa mãn. Trường thứ 4 là trường controler, trường này được sử dụng để xác định rằng các nhiệm vụ được tạo ra từ form này sẽ được xử lý bởi controler hello.

Đây là code đầy đủ của file default.php:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>
<form action="index.php" method="post" name="adminForm">
<div id="editcell">
    <table class="adminlist">
    <thead>
        <tr>
            <th width="5">
                <?php echo JText::_( 'ID' ); ?>
            </th>
            <th>
                <?php echo JText::_( 'Greeting' ); ?>
            </th>
        </tr>
    </thead>
    <?php
    $k = 0;
    for ($i=0, $n=count( $this->items ); $i < $n; $i++)
    {
        $row =& $this->items[$i];
        ?>
        <tr class="<?php echo "row$k"; ?>">
            <td>
                <?php echo $row->id; ?>
```

```
            </td>
            <td>
                <?php echo $row->greeting; ?>
            </td>
        </tr>
        <?php
```

```
        $k = 1 - $k;
```

```
    }
    ?>
    </table>
</div>

<input type="hidden" name="option" value="com_hello" />
<input type="hidden" name="task" value="" />
<input type="hidden" name="boxchecked" value="0" />
<input type="hidden" name="controller" value="hello" />
</form>
```

View hellos của chúng ta đã hoàn thành. Bạn có thể thử các kết quả của component này ngay bây giờ. Component có thể download tại:

http://dev.joomla.org/components/com_jd-wiki/data/media/components/com_hello4a.zip

4. Getting Down and Dirty: Doing the Real Work

Cho đến bây giờ view hellos đã làm việc tốt. Chúng ta sẽ chuyển sang model và view hello. Đây là nơi thao tác thực sự như add, edit, … sẽ được thực hiện.

4.1. Hello Controler

Our default controller just isn't going to cut it when it comes to doing work - all it is capable of doing is displaying views [tạm dịch Controler mặc định sẽ không ngắt giữa chừng khi nó khi nó đang làm việc- tất cả chỉ là khả năng làm hiển thị các view]. Chúng ta cần xử lý các nhiệm vụ mà chúng ta đã có ở view hellos: 'add', 'edit' và 'remove'.

'Add' và 'edit' về bản chất là hai tác vụ tương tự nhau. Cả hai đều thể hiện một form đến người dùng cho phép hiệu chỉnh một câu chào mừng. Chỉ có sự khác nhau là 'add' sẽ hiển thị một form trống, còn 'edit' sẽ hiển thị một form đã có dữ liệu của nó trong đó. Bởi vì các tác vụ giống nhau nên chúng ta sẽ ánh xạ tác vụ 'add' vào trong tác vụ 'edit'. Điều này được thực hiện trong hàm khởi tạo:

```
/**
 * constructor (registers additional tasks to methods)
 * @return void
```

```
 */
function __construct()
{
    parent::__construct();
    // Register Extra tasks
    $this->registerTask( 'add'  ,     'edit' );
}
```

Tham số đầu tiên của JControler::registerTask là tác vụ cần ánh xạ và tham số thứ hai là phương thức mà nó sẽ ánh xạ đến.

Chúng ta sẽ bắt đầu xử lý tác vụ 'edit'. Công việc của controler là khá đơn giản đối với tác vụ 'edit'. Tất cả những gì mà nó phải làm là xác định view và form layout cần tải (hello view và form layout). Ngoài ra, chúng ta sẽ nói với joomla disable main menu trong khi chúng ta hiệu chỉnh câu chào mừng. Điều này sẽ ngăn người dùng khỏi việc thoát mà quên không lưu các bản ghi đang mở.

Code cho tác vụ 'edit' của chúng ta sẽ như sau:

```
/**
 * display the edit form
 * @return void
 */
function edit()
{
    JRequest::setVar( 'view', 'hello' );
    JRequest::setVar( 'layout', 'form'  );
    JRequest::setVar('hidemainmenu', 1);
    parent::display();
}
```

4.2. Hello View

Hello view sẽ hiển thị một form cho phép người dùng hiệu chỉnh câu chào mừng. Phương thức display của view phải thực hiện một số tác vụ đơn giản sau:

- Nhận dữ liệu từ model (retrieve the data from the model)
- Tạo toolbar (create the toolbar)
- Chuyển dữ liệu vào template (pass the data into the template)
- Gọi phương thức display để trả lại template (invoke the display() method to render the template)

Việc thực hiện các tác vụ trên có một chút phức tạp, vì một view sẽ xử lý cả hai nhiệm vụ 'edit' và 'task'. Trong toolbar chúng ta muốn người dùng biết họ đang 'add' hay 'edit', bởi thế chúng ta cần phải xác định tác vụ nào được tạo ra.

Vì chúng ta đã nhận lại các bản ghi muốn hiển thị từ model, nên chúng ta có thể sử dụng dữ liệu này để xác định tác vụ nào được đưa ra. Nếu tác vụ là 'edit' thì trường 'id' sẽ được thiết lập, nếu nhiệm vụ là 'add' thì nó sẽ không được thiết lập. Điều này có thể được sử dụng để xác định xem chúng ta có một bản ghi mới hay một bản ghi đã tồn tại.

Chúng ta sẽ bổ sung thêm 2 button vào toolbar: 'save' và 'cancel'. Tuy các chức năng là giống nhau, nhưng chúng ta muốn việc hiển thị của các nút phải khác nhau, phụ thuộc vào nó là một bản ghi mới hay một bản ghi đã tồn tại. Nếu nó là bản ghi mới, chúng ta sẽ thể hiện nút 'cacel', nếu nó là bản ghi đã tồn tại, chúng ta sẽ thể hiện nút 'close'.

Vì vậy phương thức của chúng ta sẽ như sau:

```
/**
 * display method of Hello view
 * @return void
 **/
function display($tpl = null)
{
    //get the hello
    $hello        =& $this->get('Data');
    $isNew        = ($hello->id < 1);

    $text = $isNew ? JText::_( 'New' ) : JText::_( 'Edit' );
    JToolBarHelper::title(   JText::_( 'Hello' ).': <small><small>[ ' .
$text.' ]</small></small>' );
    JToolBarHelper::save();
    if ($isNew)  {
        JToolBarHelper::cancel();
    } else {
        // for existing items the button is renamed `close`
        JToolBarHelper::cancel( 'cancel', 'Close' );
    }
    $this->assignRef('hello', $hello);
    parent::display($tpl);
}
```

4.3. Hello Model

View của chúng ta cần phải có dữ liệu. Vì vậy chúng ta phải tạo ra một model để tạo ra một câu chào mừng.

Model của chúng ta sẽ có 2 thuộc tính: _id và _data. _id sẽ giữ 'id' của câu chào mừng và _data sẽ chứa dữ liệu.

Chúng ta sẽ bắt đầu với một hàm khởi tạo hướng tới việc nhận 'id' từ truy vấn:

```
/**
 * Constructor that retrieves the ID from the request
 *
 * @access    public
 * @return    void
 */
function __construct()
{
    parent::__construct();
```

```
    $array = JRequest::getVar('cid',  0, '', 'array');
    $this->setId((int)$array[0]);
}
```

Phương thức JRequest::getVar() được sử dụng để nhận dữ liệu từ request. Tham số đầu tiên là tên của biến form. Tham số thứ hai là giá trị mặc định sẽ được gán nếu không có giá trị nào được tìm thấy. Tham số thứ 3 là tên của hash để nhận dữ liệu từ (get, post, vv) và giá trị cuối cùng là kiểu dữ liệu của giá trị.

Hàm khởi tạo của chúng ta sẽ lấy giá trị đầu tiên từ mảng 'cid' và gán nó cho 'id'.

Phương thức setId() được sử dụng để thiết lập 'id'. Việc thay đổi 'id' mà model của chúng ta trỏ tới có nghĩa là dữ liệu mà 'id' trỏ tới sẽ không còn đúng nữa. Do đó, khi chúng ta thiết lập 'id' chúng ta cần clear thuộc tính data.

```
/**
 * Method to set the hello identifier
 *
 * @access    public
 * @param    int Hello identifier
 * @return    void
 */
function setId($id)
{
    // Set id and wipe data
    $this->_id        = $id;
    $this->_data    = null;
}
```

Cuối cùng, chúng ta cần một phương thức để nhận dữ liệu: getData().

Phương thức getData sẽ kiểm tra xem thuộc tính data đã được thiết lập hay chưa. Nếu nó đã được thiết lập thì getData sẽ trả lại nó, nếu không getData sẽ tải dữ liệu từ CSDL.

```
**
 * Method to get a hello
 * @return object with data
 */

function &getData()
{
    // Load the data
```

```
    if (empty( $this->_data )) {
```

```
        $query = ' SELECT * FROM #__hello '.
                '  WHERE id = '.$this->_id;
        $this->_db->setQuery( $query );
        $this->_data = $this->_db->loadObject();
    }
    if (!$this->_data) {
        $this->_data = new stdClass();
        $this->_data->id = 0;
        $this->_data->greeting = null;
    }
    return $this->_data;
}
```

4.4. The form

Bây giờ, công việc còn lại là tạo ra form để đưa dữ liệu vào. Vì chúng ta đã xác định layout của chúng ta như một form. Form sẽ được để trong một file trong thư mục tmp của hello view gọi là form.php:

```
<?php defined('_JEXEC') or die('Restricted access'); ?>

<form action="index.php" method="post" name="adminForm" id="adminForm">
<div class="col100">
    <fieldset class="adminform">
        <legend><?php echo JText::_( 'Details' ); ?></legend>
        <table class="admintable">
        <tr>
            <td width="100" align="right" class="key">
```

```
                <label for="greeting">
                    <?php echo JText::_( 'Greeting' ); ?>:
                </label>
            </td>
            <td>
                <input class="text_area" type="text" name="greeting"
id="greeting" size="32" maxlength="250" value="<?php echo $this->hello-
>greeting;?>" />
            </td>
        </tr>
    </table>
    </fieldset>
</div>
<div class="clr"></div>
<input type="hidden" name="option" value="com_hello" />
<input type="hidden" name="id" value="<?php echo $this->hello->id; ?>"
/>
<input type="hidden" name="task" value="" />
<input type="hidden" name="controller" value="hello" />
</form>
```

Chú ý rằng trong việc bổ sung trường input, có một trưởng hidden input cho 'id'. Người dùng sẽ không cần hiệu chỉnh 'id', nên chúng ta sẽ không thể hiện nó trên form.

4.5. Cài đặt các chức năng

Cho đến lúc này controler của chúng ta chỉ thực thi hai nhiệm vụ: edit và new. Tuy nhiên chúng ta còn có các nút để 'save', 'delete', và 'cancel' các bản ghi. Chúng ta cần viết các hàm để xử lý và thực hiện các tác vụ này.

4.5.1. Lưu bản ghi

Bước tiếp theo là cài đặt chức năng để lưu bản ghi. Thông thường, cần phải có một vài sự thay đổi để xử lý các trường hợp khác nhau, như sự khác nhau giữa việc tạo bản ghi mới (câu lệnh INSERT) và cập nhật bản ghi đã tồn tại (câu lệnh UPDATE). Ngoài ra rất phức tạp để có thể lấy dữ liệu từ form và đưa nó vào câu truy vấn.

Tuy nhiên, Joomla! Framework đã chuẩn bị việc này khá cẩn thận cho bạn. Lớp JTable giúp việc thao tác dễ dàng trên các bản ghi trên CSDL mà không phải lo lắng về việc viết các lệnh SQL nằm bên dưới. Nó còn giúp việc chuyển dữ liệu từ HTML vào CSDL được dễ dàng.

a) Tạo lớp table

Lớp table là một lớp trừu tượng từ đó bạn có thể tạo ra các lớp con để làm việc với các bảng cụ thể. Để thực hiện điều đó đơn giản là chúng ta tạo ra một lớp kế thừa lớp

JTable, bổ sung các trường dữ liệu như các thuộc tính và ghi đè hàm khởi tạo để xác định tên của bảng và khóa chính.

Dưới đây là lớp JTable của chúng ta:

```
<?php
/**
 * Hello World table class
 * @package    Joomla.Tutorials
 * @subpackage Components
 * @link http://dev.joomla.org/component/option,com_jd-
wiki/Itemid,31/id,tutorials:components/
 * @license        GNU/GPL
 */
// no direct access
defined('_JEXEC') or die('Restricted access');
/**
 * Hello Table class
 *
 * @package    Joomla.Tutorials
 * @subpackage Components
 */
class TableHello extends JTable
{
    /**
     * Primary Key
     *
     * @var int
     */
    var $id = null;

    /**
     * @var string
     */
    var $greeting = null;

    /**
     * Constructor
     *
     * @param object Database connector object
     */
    function TableHello( &$db ) {
```

```
        parent::__construct('#__hello', 'id', $db);
    }
}
?>
```

Ở đây bạn sẽ thấy, chúng ta đã định nghĩa hai trường: trường 'id' và trường 'greeting'. Sau đó chúng ta định nghĩa một hàm khởi tạo, nó sẽ gọi hàm khởi tạo của lớp cha và truyền vào đó tên của bảng (hello), tên của trường là khóa chính (id) và đối tượng kết nối CSDL.

File này sẽ được gọi hello.php và nó sẽ nằm trong thư mục table trong phần quản trị của component.

b) Cài đặt chức năng trong model

Bây giờ chúng ta đã sẵn sàng bổ sung phương thức vào model để nó lưu lại bản ghi. Chúng ta sẽ gọi phương thức này để lưu trữ. Phương thức store() sẽ thực hiện 3 việc: kết khối dữ liệu từ form vào đối tượng TableHello, kiểm tra để chắc chắn rằng bản ghi được định dạng đúng, và lưu dữ liệu vào CSDL.

Phương thức của chúng ta như sau:

```
/**
 * Method to store a record
 *
 * @access    public
 * @return    boolean    True on success
 */
function store()
{
    $row =& $this->getTable();

    $data = JRequest::get( 'post' );
    // Bind the form fields to the hello table
    if (!$row->bind($data)) {
        $this->setError($this->_db->getErrorMsg());
        return false;
    }

    // Make sure the hello record is valid
    if (!$row->check()) {
        $this->setError($this->_db->getErrorMsg());
        return false;
    }
```

```
    // Store the web link table to the database
    if (!$row->store()) {
        $this->setError($this->_db->getErrorMsg());
        return false;
    }
    return true;
}
```

Phương thức này được bổ sung vào model hello.

Phương thức sẽ có một tham số. Nó là một mảng liên kết dữ liệu mà chúng ta muốn lưu trữ vào CSDL. Điều này có thể được thực hiện dễ dàng từ câu truy vấn sẽ được xem xét trong phần sau

Chúng ta thấy rằng dòng đầu tiên trả lại một tham chiếu đến đối tượng JTable. Nếu chúng ta đặt tên table chính xác chúng ta sẽ không phải xác định tên của nó - lớp JModel sẽ biết tự tìm nó ở đâu. Nhắc lại rằng lớp table của chúng ta được gọi là TableHello và đặt trong file hello.php nằm trong thư mục tables. Nếu bạn tuân theo quy luật này lớp JModel có thể tạo ra đối tượng của bạn một cách tự động.

Dòng thứ hai sẽ nhận lại dữ liệu từ form. Lớp JRequest sẽ thực hiện điều này rất dễ dàng. Trong trường hợp này chúng ta sẽ nhận lại tất cả các biến mà chúng ta submit sử dụng phương thức POST. Điều này sẽ trả lại một mảng liên kết.

Phần còn lại là rất dễ dàng. Chúng ta gán dữ liệu, kiểm tra và lưu trữ. Phương thức bind() sẽ copy dữ liệu từ mảng vào thuộc tính tương ứng của đối tượng table. Trong trường hợp này, phương thức bind() sẽ lấy giá trị của ‘id’ và ‘greeting’ sau đó copy chúng vào các thuộc tính của đối tượng table.

Phương thức store() lấy dữ liệu từ đối tượng và lưu nó vào CSDL. Nếu id =0 nó sẽ tạo ra một bản ghi mới (câu lệnh insert), nếu không nó sẽ cập nhật bản ghi đã có (câu lệnh update).

c) Bổ sung tác vụ vào controler

Bây giờ chúng ta sẽ bổ sung tác vụ vào controler. Vì tác vụ mà chúng ta đang thuwcj hiện là ‘save’, nên chúng ta phải gọi phương thức save:

```
/**
 * save a record (and redirect to main page)
 * @return void
 */
function save()
{
    $model = $this->getModel('hello');

    if ($model->store()) {
```

```
        $msg = JText::_( 'Greeting Saved!' );
    } else {
        $msg = JText::_( 'Error Saving Greeting' );
    }

    // Check the table in so it can be edited.... we are done with it
anyway
    $link = 'index.php?option=com_hello';
    $this->setRedirect($link, $msg);
}
```

Tất cả những gì chúng ta làm là lấy model và gọi phương thức store. Sau đó chúng ta sử dụng phương thức setRedirect() để redirect trở lại danh sách các câu chào mừng. Ngoài ra chúng ta còn truyền cho nó một message, message này sẽ được hiển thị ở phía trên đầu của trang.

4.5.2. Xóa một bản ghi

a) Cài đặt chức năng trong Model

Trong model chúng ta sẽ nhận lại một danh sách các 'id' cần xóa và gọi lớp JTable để xóa chúng.

```
/**
 * Method to delete record(s)
 * @access    public
 * @return    boolean    True on success
 */
function delete()
{
    $cids = JRequest::getVar( 'cid', array(0), 'post', 'array' );
    $row =& $this->getTable();

    foreach($cids as $cid) {
        if (!$row->delete( $cid )) {
            $this->setError( $row->getErrorMsg() );
            return false;
        }
    }
    return true;
}
```

Chúng ta gọi phương thức JRequest::getVar() để lấy dữ liệu từ request sau đó gọi phương thức $row→delete() để xóa mỗi dòng. Bằng việc lưu trữ các lỗi trong model chúng ta sẽ có thể nhận lại chúng sau đó nếu chúng ta muốn.

b) Xử lý tác vụ remove trong controler

Tương tự như phương thức save() trong việc xử lý tác vụ lưu trữ.

```
**
 * remove record(s)
 * @return void
 */
function remove()
{
    $model = $this->getModel('hello');
    if(!$model->delete()) {
        $msg = JText::_( 'Error: One or More Greetings Could not be
Deleted' );
    } else {
        $msg = JText::_( 'Greeting(s) Deleted' );
    }

    $this->setRedirect( 'index.php?option=com_hello', $msg );
}
```

4.5.3. Hủy bỏ quá trình hiệu chỉnh

Để hủy bỏ việc ‘edit’, tất cả những gì chúng ta cần làm là redirect lại main view.

```
/**
 * cancel editing a record
 * @return void
 */
function cancel()
{
    $msg = JText::_( 'Operation Cancelled' );
    $this->setRedirect( 'index.php?option=com_hello', $msg );
}
```

5. Kết luận

We have now implemented a basic backend to our component. We are now able to edit the entries that are viewed in the frontend. We have demonstrated the interaction

between models, views and controllers. We have shown how the JTable class can be extended to provide easy access to tables in the database. It can also be seen how the JToolBarHelper class can be used to create button bars in components to present a standardized look between components. ☺.

6. Những người viết và địa chỉ download ví dụ

Những người tham gia viết bài hướng dẫn này:

- staalanden

Component có thể được download tại:
http://dev.joomla.org/components/com_jd-wiki/data/media/components/com_hello4.zip