



# Using Ruby on Rails Sites with WordPress MU Blogs

This white paper talks about how to get WordPress MU blogs to work with Ruby on Rails sites. It addresses problem scenarios, approaches, and security.



## WordPress MU (<http://mu.wordpress.org/>)

WordPress is a state-of-the-art publishing platform with a focus on aesthetics, web standards, and usability. WordPress is both free and priceless at the same time. It is the most popular web and blogging tool in existence, and there are thousands of free plug-ins and themes available for use. SEO friendliness is another powerful feature of WordPress. WordPress MU (“Multi-User”) is designed to scale millions of page views and unlimited users and blogs.

## Ruby on Rails (<http://rubyonrails.org/>)

Ruby on Rails, often shortened to “Rails” or “RoR”, is an open source web application framework for the Ruby programming language. It is intended to be used with an agile development methodology, which is used by web developers for rapid development.

## Scenario

We built a high-traffic social community, which offered a niche product to provide information about maintaining a healthy lifestyle, including articles on various health topics. This application was developed using Ruby on Rails, and there were many advanced requirements for the article pages of the site. Since the article pages (analogous to blogs) were not the “meat” of the product, we decided to implement the WordPress MU tool, which provides an excellent platform for multi-user blogging.

## Integrating WordPress MU with a Rails Site

To integrate the WordPress MU tool, you need to place WordPress project under Rails project's “Public” directory and write a custom apache rewrite rule:

```
RewriteCond %{HTTP_HOST} ^YOURDOMAIN.com [NC]
RewriteRule ^/wordpress/?(.*)$ /<RAILS_ROOT>/public/wordpress/$1 [QSA,NC,L]
```

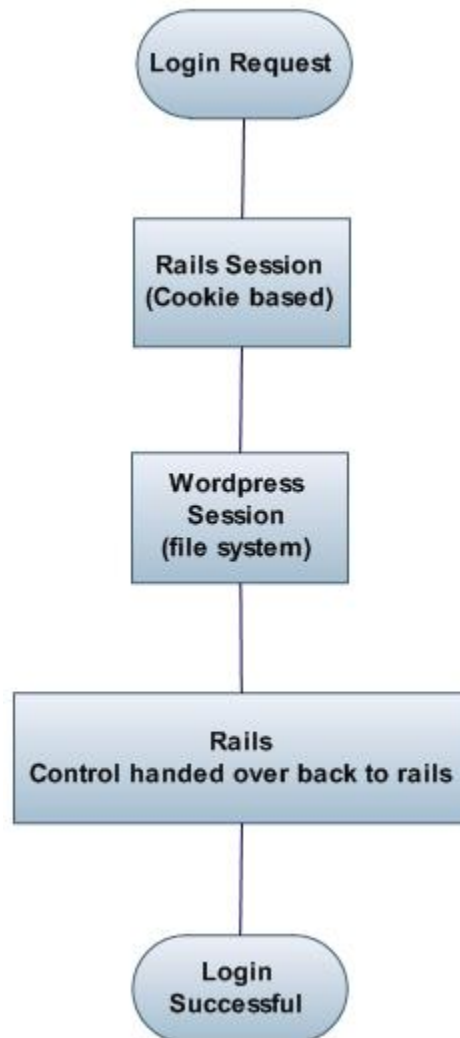
## Sharing Users and Sessions

Upon successful authentication, the Rails server creates a session cookie. WordPress MU has its own authentication process, but to keep the user experience transparent, you can manage the WordPress authentication through the back-end. Please read below to learn how to implement a single sign-on:

Add following code into the index.php file from WordPress Theme Directory:

```
# Storing session for rails user
if ($_REQUEST['user_id'])
{
    session_start();
    $_SESSION['gbl_user_id']=$_REQUEST['user_id'];
}
```

This code creates the "user\_id" session in WordPress.



To check the user sessions, you will need to add a custom function:

#checking user sessions through PHP

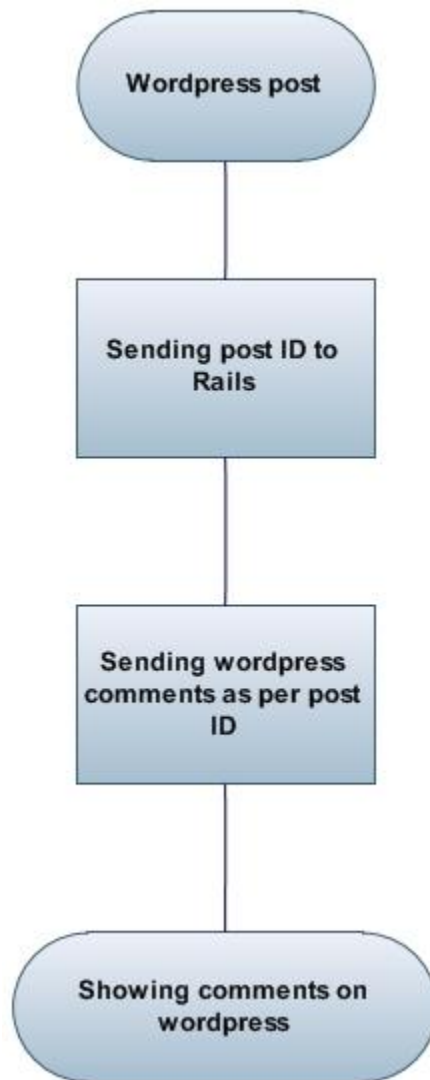
```
function check_session()
{
    ob_start();
    session_start();
    if (!isset($_SESSION['gbl_user_id']))
    {
        header("location:");
    }
}
```



## Commenting on WordPress MU blogs

For users to comment on the articles, they must be registered and stored in the Rails site database, and the articles that will allow comments must be specified as such. These users need to be identified on the article page, which is served via WordPress MU. In order to achieve this, Rails must be set up to identify the articles on which users can comment.

We created MySQL triggers on `wp_posts`, `wp_terms`, `wp_term_taxonomy` WordPress MU tables. These triggers duplicate the articles (whenever an article is published or updated) to the Rails database (along with `post_id`). This `post_id` is then used by the Rails coder, in conjunction with the `acts_as_commentable` plug-in, to manage the relationship between comments and posts (articles).



In order to display the existing comments on WordPress MU blogs, the comments are “fetched” from the Rails database based on the ‘post\_id’ and the ‘user\_id’. All the processing is handled by Rails; all we had to do was display the HTML elements necessary for commenting in an iFrame on the WordPress MU pages.

## Security

Our requirement was to use authentication for the Rails pages and WordPress MU blogs. However, the user data was stored in the Rails database, and we needed the same user credentials to be used on both Rails site and WordPress MU blog.

The main security concern was regarding the sessions used by both the Rails site and the WordPress MU blog. If we used a common session for both, there was a risk of losing important session data, as well as the chance of occasionally handling tampered cookies.

So we decided to use different sessions with the same credentials for Rails and WordPress MU. For Rails, we use a cookie-based session, while for WordPress MU we use the default file-system based session.

Whenever an end-user logs into the site, a session is created in Rails first. The control is then passed onto WordPress MU, and a session is created for the WordPress MU blog. Once the session is created, the control is again handed over to Rails, with the process going in reverse for logging out of the sessions.

## Configurations

We have a standard WordPress MU Installation with the WordPress MU version being 2.6. Rails site has a mongrel\_cluster with multiple mongrel instances all behind a standard Apache installation version 2.2.

We used MySQL database version 5.0. We built separate databases for Rails and WordPress MU. Configuration for the Rails database is specified in the <RAILS\_ROOT>/config/database.yml file, whereas the configuration for WordPress MU database is specified in the <WORDPRESS\_ROOT>/wp-config.php file.

## Building a site page with contents served by both Rails and WordPress MU

The home page on the Rails site has videos, images, and highlights of articles from WordPress MU Blogs, in addition to the Rails site content.

In order to achieve this, we created a Rails page that had the Rails content, and we embedded iFrame(s) for the WordPress MU Blogs content. We created a custom script in WordPress MU for showing videos, images, and articles to handle the requests from the iFrame(s).