



Core Concepts & Configuration

Key Concepts

HMVC (Hierarchical Model-View-Controller):
FuelPHP extends the traditional MVC pattern to HMVC, promoting modularity and reusability of code.

ORM (Object-Relational Mapper): Provides an ActiveRecord implementation for easy database interaction.

Security: Built-in CSRF protection, input filtering, and output encoding to prevent common web vulnerabilities.

Bundles: Reusable packages of code that can be easily integrated into FuelPHP applications.

Modules: Self-contained applications within a FuelPHP project, enabling code organization and separation of concerns.

Configuration Files

<code>config.php</code>	Main application configuration file. Located in <code>fuel/app/config/</code> .
<code>routes.php</code>	Defines URL routes. Located in <code>fuel/app/config/</code> .
<code>database.php</code>	Database connection settings. Located in <code>fuel/app/config/</code> .
<code>autoload.php</code>	Specifies classes and packages to automatically load. Located in <code>fuel/app/config/</code> .

Environment Configuration

FuelPHP supports environment-specific configurations (development, production, testing).
Configuration files are loaded in the following order:

- `fuel/app/config/config.php`
- `fuel/app/config/{environment}/config.php` (overrides defaults)

Controllers, Models & Views

Controllers

Controllers handle user requests and interact with models to retrieve or modify data. They then pass data to views for rendering.

Example:

```
class Controller_Users extends Controller
{
    public function action_index()
    {
        $data['users'] = Model_User::find_all();
        return View::forge('users/index', $data);
    }
}
```

Models

Models represent data and provide methods for interacting with the database. FuelPHP uses an ActiveRecord ORM.

Example:

```
class Model_User extends Orm\Model
{
    protected static $_properties = array(
        'id',
        'username',
        'password',
        'email',
    );
}
```

Views

Views are responsible for rendering data provided by controllers into HTML or other formats.

Example (`fuel/app/views/users/index.php`):

```
<h1>Users</h1>
<ul>
<?php foreach ($users as $user): ?>
    <li><?php echo $user->username; ?>
</li>
<?php endforeach; ?>
</ul>
```

Use `View::forge()` to create a view instance in your controller.

Routing & URI Handling

Basic Routing

Routes define how URLs are mapped to controllers and actions. Defined in `fuel/app/config/routes.php`.

Example:

```
return array(
    '_root_' => 'welcome/index', //
    // The default route
    '_404_' => 'welcome/404', //
    // The main 404 route
    'hello/(:any)' =>
    array('welcome/hello', 'name' => '$1'),
);
```

Named Parameters

You can use named parameters in your routes. Example:

```
'users/:id' => 'users/view/$id'
```

URI Class

<code>URI::base()</code>	Returns the base URL of the application.
<code>URI::current()</code>	Returns the current URI.
<code>URI::segment(\$n)</code>	Returns the nth segment of the URI.

ORM & Database

Basic ORM Usage

FuelPHP's ORM simplifies database interactions. Remember to configure your database settings in <code>fuel/app/config/db.php</code> .
Example (Retrieving Data): <pre>\$user = Model_User::find(1); echo \$user->username;</pre>
Example (Creating Data): <pre>\$user = Model_User::forge(array('username' => 'newuser', 'password' => 'password123', 'email' => 'newuser@example.com',)); \$user->save();</pre>

Relationships

FuelPHP supports various relationship types (one-to-one, one-to-many, many-to-many).
Example (One-to-Many in Model_User): <pre>protected static \$_has_many = array('posts' => array('model_to' => 'Model_Post', 'key_from' => 'id', 'key_to' => 'user_id',));</pre>
Now you can access the user's posts: <pre>\$user = Model_User::find(1); foreach (\$user->posts as \$post) { echo \$post->title; }</pre>

Query Builder

For more complex queries, you can use the Query Builder.
Example: <pre>\$query = DB::select('id', 'username') ->from('users') ->where('username', 'like', '%admin%') ->order_by('id', 'desc') ->limit(10); \$result = \$query->execute(); foreach (\$result as \$row) { echo \$row['username']; }</pre>