PWO

PHP Web Objects (PWO) — Worker

Simple background worker for AI multimodal processing, document workflows, and SMTP mailing · converted: 2026-01-13

The PHP Web Objects (PWO) Worker is a high-performance background CLI process designed to handle heavy lifting — such as AI multimodal processing, automated document workflows, and SMTP mailing — with resilience and efficient resource management.

1. Linux Setup (Production with Supervisor)

In a production environment, run the worker as a daemon. Supervisor is recommended.

Install Supervisor

```
sudo apt-get install supervisor
```

Configure the Worker

Create a configuration file: `/etc/supervisor/conf.d/pwo-worker.conf`

```
[program:pwo-worker]
process_name=%(program_name)s_%(process_num)02d
command=php /var/www/your-project/app/worker.php
autostart=true
autorestart=true
user=www-data
numprocs=1
redirect_stderr=true
stdout_logfile=/var/www/your-project/logs/worker.log
```

Start the Service

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start pwo-worker:*
```

2. Windows Setup (Production with NSSM)

Windows needs a service manager. Use NSSM (Non-Sucking Service Manager).

1. Download: NSSM.cc

Source: myapp/app/README.md Generated: 2026-01-13

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start pwo-worker:*
```

(Note: those NSSM examples reference Supervisor commands in your original README — replace with appropriate NSSM commands or use the NSSM GUI.)

3. Configuration:
   - Path: `C:\php\php.exe`
   - Startup Directory: `C:\path\to\your\project\app`
   - Arguments: `worker.php`
4. Registry: Click "Install Service" and start it via `services.msc`.

3. Worker Lifecycle & Performance

The PWO Worker is built with "Zero-Waste" logic:

- **Memory Management:** The worker monitors its own memory usage. If it exceeds 64MB (configurable), it exits gracefully. Supervisor/NSSM restarts it with a clean process.
- **Exponential Backoff:** On task failures (AI API calls, SMTP sends), the worker updates the `available_at` time. Retries occur at 5 minutes, 15 minutes, and 1 hour intervals.
- **Catch-All Resilience:** Uses `catch(\Throwable)` to ensure that even handler errors are logged to the database and do not crash the whole service.

4. Adding a New Job Handler

Follow the OOP pattern to extend the worker:

1. Create a class `QueueNewClass` in `app/queues/QueueNewClass.php` implementing `JobHandlerInterface`.
2. Register the task name in the `QueueWorker` constructor:

```
$this->handlers['your_task_name'] = new QueueNewClass();
```

> Tip: For quick PDF export, open this file in Chrome and press **Ctrl/Cmd+P** → "Save as PDF". For automated conversion, use `wkhtmltopdf README-styled.html README.pdf` or `pandoc README.md -o README.pdf`.

Source: myapp/app/README.md Generated: 2026-01-13