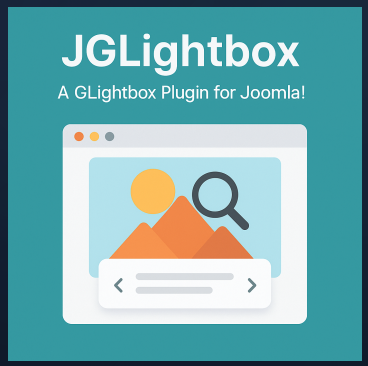
Jglightbox
Plugin-Handbuch

Dieses Dokument beschreibt Aufbau, Konfiguration und Nutzung des Joomla Content Plugins **Jglightbox**. Das Plugin bindet die GLightbox-Library ein und stellt Optionen zur Steuerung der Lightbox-Aktivierung bereit (Wrapper-Klasse & Exclude-Selektoren).



Paket Joomla.Plugin / content

Namespace Joomla\Plugin\Content\Jglightbox\Extension

Autor jumbo125

Lizenz GNU GPL v2+

Inhaltsverzeichnis

Was macht das Plugin?

Beim Joomla-Event `onBeforeCompileHead` registriert und lädt das Plugin CSS/JS über den **WebAssetManager** und übergibt Konfiguration per `addScriptOptions()` an JavaScript.

> **Kernidee:** Du definierst, welche Elemente als Lightbox „aktiv" gelten (Wrapper-Klasse) und welche Elemente explizit ausgeschlossen werden (CSS-Selektoren).

Baustein	Aufgabe
`Jglightbox.php`	Event-Subscriber: Assets laden & Script-Optionen setzen
`services/provider.php`	DI-Registrierung: Plugin instanziieren, Parameter

	setzen, Application injizieren
`joomla.asset.json`	Definition der Web Assets (Styles/Scripts) unter `media/plg_content_jglightbox/`

1. Quickguide

1.1 Installation

1. Plugin-Paket als Joomla-Erweiterung installieren (ZIP).
2. Plugin unter **Erweiterungen** → **Plugins** suchen: `Content - Jglightbox`.
3. Plugin aktivieren.

1.2 Minimale Konfiguration

- **Wrapper-Klasse** (Standard: `lightbox_wrapper`)
- **Exclude-Selektoren** (Standard: `.slides img,.delete,.edit-icon img,.no-lightbox`)

> **Empfehlung:** Gib Bildern/Galerien, die in die Lightbox sollen, einen gemeinsamen Wrapper (z. B. `lightbox_wrapper`) und markiere Ausnahmen mit `.no-lightbox`.

1.3 Beispiel (HTML im Beitrag)

```
<div class="lightbox_wrapper">
  <a href="/images/full/1.jpg"><img src="/images/thumb/1.jpg" alt=""></a>
  <a href="/images/full/2.jpg"><img src="/images/thumb/2.jpg" alt=""></a>
</div>
```

```
<!-- ausgeschlossen -->
<img class="no-lightbox" src="/images/thumb/no.jpg" alt="">
```

2. Überblick

Das Plugin basiert auf der GLightbox-Library (Biati Digital, MIT). Es nutzt Joomla 4/5 typische Muster: Event-Subscriber (`SubscriberInterface`) und den WebAssetManager mit Registry-Datei.

2.1 Ablauf in Kurzform

1. Joomla feuert `onBeforeCompileHead` .
2. Plugin lädt Asset-Registry (`joomla.asset.json`).
3. Plugin aktiviert definierte Styles/Scripts via `useStyle()` / `useScript()` .
4. Plugin übergibt Konfiguration an JS via `addScriptOptions('jglightbox', ...)` .

3. Installation & Aktivierung

3.1 Erwartete Verzeichnisstruktur (typisch)

```
plugins/content/jglightbox/
  ├─ src/Extension/Jglightbox.php
  ├─ services/provider.php
  └─ jglightbox.xml  (Manifest, nicht im Snippet enthalten)

media/plg_content_jglightbox/
  ├─ joomla.asset.json
  ├─ css/...
  └─ js/...
```

Wichtig: Das Plugin setzt voraus, dass die Asset-Registry unter `media/plg_content_jglightbox/joomla.asset.json` existiert und die

dort definierten Handles (`glightbox_style` , `glightbox_original` , `glightbox_original_inject`) korrekt benannt sind.

4. Konfiguration (Parameter)

Parameter	Standard	Bedeutung
`wrapper_classes`	`lightbox_wrapper`	CSS-Klasse(n), die als „Container" für Lightbox-Inhalte dienen. In JavaScript wird typischerweise innerhalb dieses Wrappers nach passenden Links/Bildern gesucht.
`exclude_selectors`	`.slides img,.delete,.edit-icon img,.no-lightbox`	CSS-Selektoren, die von der Lightbox ausgenommen werden (z. B. Admin-Icons, Slider-Images, manuelle Ausnahmen).

Praxis-Tipp: Wenn du unerwartete Lightbox-Auslösungen siehst, erweitere `exclude_selectors` um passende Selektoren aus deinem Template (z. B. Slider/Galerie-Module).

5. Funktionsweise & Events

5.1 Event-Subscription

Das Plugin implementiert `SubscriberInterface` und registriert sich für `onBeforeCompileHead` . Dadurch werden Assets sehr früh in den Dokument-Head eingebunden, bevor Joomla den Head final kompiliert.

5.2 Was passiert in `onBeforeCompileHead` ?

1. WebAssetManager beziehen: `Factory::getApplication()->getDocument()->getWebAssetManager()`
2. Asset-Registry hinzufügen: `addRegistryFile(.../joomla.asset.json)`
3. Assets aktivieren: `useStyle` / `useScript`
4. Parameter lesen und via `addScriptOptions` an JavaScript übergeben

> **Hinweis:** In der vorliegenden Implementierung werden Assets auf jeder Seite geladen, auf der das Event ausgeführt wird. Wenn du das nur auf bestimmten Seiten möchtest, wäre eine Erweiterung sinnvoll (z. B. nur laden, wenn Content bestimmte Marker/Wrapper enthält).

6. Assets & WebAssetManager

Joomla verwaltet CSS/JS über den WebAssetManager. Die Asset-Definitionen liegen in einer Registry-Datei (`joomla.asset.json`). Das Plugin aktiviert dann die dort registrierten Handles.

6.1 Verwendete Handles

- `glightbox_style` – CSS für GLightbox
- `glightbox_original` – Original-GLightbox Script
- `glightbox_original_inject` – Zusatzscript (vermutlich Initialisierung + Lesen der ScriptOptions)

> **Wichtig für JS:** Die Konfiguration steht clientseitig typischerweise unter `Joomla.getOptions('jglightbox')` oder über den Joomla ScriptOptions-Mechanismus zur Verfügung (abhängig von deiner Initialisierungsdatei).

7. Verwendung im Content

7.1 Standard-Workflow

1. Setze einen Wrapper um den Bereich, der Lightbox-fähig sein soll: `lightbox_wrapper` .
2. Verwende anklickbare Links auf große Bilder (klassische Lightbox-Struktur).
3. Schließe unerwünschte Elemente per `exclude_selectors` oder `.no-lightbox` aus.

7.2 Beispiel: Galerie

```
<div class="lightbox_wrapper">
  <figure>
```

```
    <a href="/images/large/a.jpg">
      <img src="/images/thumb/a.jpg" alt="Bild A">
    </a>
    <figcaption>Bild A</figcaption>
  </figure>

  <figure>
    <a href="/images/large/b.jpg">
      <img src="/images/thumb/b.jpg" alt="Bild B">
    </a>
    <figcaption>Bild B</figcaption>
  </figure>
</div>
```

> **Ergebnis:** Klick auf ein Thumbnail öffnet die Lightbox (sofern dein Inject/Init-Script die Links im Wrapper verarbeitet).

8. Troubleshooting

8.1 Assets werden nicht geladen

- Prüfe, ob `media/plg_content_jglightbox/joomla.asset.json` vorhanden ist.
- Prüfe, ob die Handles in der JSON exakt so heißen wie im PHP: `glightbox_style`, `glightbox_original`, `glightbox_original_inject`.
- Browser-Konsole: 404/500 auf CSS/JS?

8.2 Lightbox reagiert auf „falsche" Bilder

- Wrapper enger setzen (nicht um den kompletten Artikel, sondern gezielt um die Galerie).
- `exclude_selectors` erweitern (z. B. Slider/Module/Editor-Icons).
- Einzelne Elemente mit `.no-lightbox` kennzeichnen.

8.3 Konflikte mit Template / JS

- Teste ohne JS-Minifier/Optimizer (Cache leeren).

- Prüfe, ob andere Lightbox-Skripte parallel geladen werden.
- Reihenfolge im Asset-Manager: ggf. Abhängigkeiten in der Asset-JSON definieren.

9. Drittanbieter-Lizenzen

> **GLightbox:** Original library: MIT License, Copyright (c) 2018 Biati Digital (laut Header-Kommentar im Code).

Stelle sicher, dass du die Lizenztexte der Drittanbieter-Bibliotheken in deinem Distributionspaket bzw. in der Dokumentation berücksichtigst (je nach Projektanforderung).

10. Anhang (Code aus dem Projekt)

10.1 Plugin-Klasse `Jglightbox` (src/Extension/Jglightbox.php)

```
<?php
/**
 * @package     Joomla.Plugin
 * @subpackage  [PLUGIN_NAME]
 * @author      jumbo125
 * @copyright   Copyright (C) 2025 jumbo125. All rights reserved.
 * @license     GNU General Public License version 2 or later; see LICENSE.txt
 *
 * Fremde Skripte / Third-party libraries:
 * - Original library: MIT License Copyright (c) 2018 Biati Digital https://www.biati.digital
 */

// Sicherheitscheck
namespace Joomla\Plugin\Content\Jglightbox\Extension;

\defined('_JEXEC') or die;

use Joomla\CMS\Plugin\CMSPlugin;
use Joomla\CMS\Factory;
use Joomla\Event\Event;
use Joomla\Event\SubscriberInterface;
```

```
use Joomla\Event\DispatcherInterface;

class Jglightbox extends CMSPlugin implements SubscriberInterface
{
    public static function getSubscribedEvents(): array
    {
        return [
            'onBeforeCompileHead' => 'onBeforeCompileHead',
        ];
    }

    public function onBeforeCompileHead(Event $event): void
    {
        $wa = Factory::getApplication()->getDocument()->getWebAssetManager();
        $wa->getRegistry()->addRegistryFile('media/plg_content_jglightbox/joomla.asset.json');

        $wa->useStyle('glightbox_style');
        $wa->useScript('glightbox_original');
        $wa->useScript('glightbox_original_inject');

        $wrapper = $this->params->get('wrapper_classes', 'lightbox_wrapper');
        $exclude = $this->params->get('exclude_selectors', '.slides img,.delete,.edit-icon img,.no-lightbox');

        Factory::getDocument()->addScriptOptions('jglightbox', [
            'wrapper' => $wrapper,
            'exclude' => $exclude,
        ]);
    }
}
```

10.2 Service Provider (services/provider.php)

```
<?php
/**
 * @package     Joomla.Plugin
 * @subpackage  [PLUGIN_NAME]
 * @author      jumbo125
 * @copyright   Copyright (C) 2025 jumbo125. All rights reserved.
 * @license     GNU General Public License version 2 or later; see LICENSE.txt
```

```
 *
 * Fremde Skripte / Third-party libraries:
 * - Original library: MIT License Copyright (c) 2018 Biati Digital https://www.biati.digital
 */

// Sicherheitscheck
defined('_JEXEC') or die;

use Joomla\CMS\Factory;
use Joomla\CMS\Extension\PluginInterface;
use Joomla\CMS\Plugin\PluginHelper;
use Joomla\Event\DispatcherInterface;
use Joomla\DI\Container;
use Joomla\DI\ServiceProviderInterface;
use Joomla\Plugin\Content\Jglightbox\Extension\Jglightbox;
use Joomla\Registry\Registry;

return new class implements ServiceProviderInterface {
    public function register(Container $container) {
        $container->set(
            PluginInterface::class,
            function (Container $container) {
                $config = (array) PluginHelper::getPlugin('content', 'jglightbox');
                $dispatcher = $container->get(DispatcherInterface::class);

                // Plugin erzeugen
                $plugin = new Jglightbox($dispatcher, $config);

                // Plugin-Parameter manuell setzen
                $plugin->params = new Registry($config['params'] ?? []);

                // Joomla Application zuweisen
                $plugin->setApplication(Factory::getApplication());

                return $plugin;
            }
        );
    }
};
```

Hinweis zur DI: Hier wird das Plugin manuell instanziiert und Parameter werden explizit in ein `Registry`-Objekt überführt. Dadurch sind `$this->params->get(...)` Aufrufe in der Plugin-Klasse möglich.