ItkRobotsTxt

Entwicklerdokumentation

André Fischer

30.03.2026

Inhaltsverzeichnis

1 Einordnung

- **Shopware-Plugin-Klasse:** `ItkComputerGmbh\\ItkRobotsTxt\\ItkRobotsTxt`
- `PSR-4-Namespace`: **`ItkComputerGmbh\\ItkRobotsTxt\\`** → `src/`
- **Abhängigkeit:** `shopware/core` ~6.6.0

Das Plugin ist bewusst **minimal**: ein Storefront-Controller, eine System-Config, DI in `services.xml`. Es gibt **keine** eigenen Entities, Migrationen, Commands oder Subscriber.

Wichtig: Eine bereits vorhandene robots.txt welche unter /public liegt muss entfernt werden. Die Datei hat eine höhere Priorität als der Controller und würde somit den Controller überschreiben.

1.1 Diagramm: Anfrage robots.txt

2 Symfony / Shopware in einem Absatz

Shopware baut auf Symfony auf. Ein **Controller** ist eine PHP-Klasse mit Methoden, die über **Routen-Attribute** (**#[Route]**) URLs zugeordnet werden. Der **Dependency Injection Container** erzeugt den Controller und injiziert z. B. den **SystemConfigService**. `SystemConfigService::get($key, $salesChannelId)` liest Werte aus der Tabelle der Plugin-Konfiguration – dieselben, die du im Admin unter Plugin-Einstellungen siehst. **SalesChannelContext** enthält die ID des aktuellen Kanals, damit die Ausgabe pro Domain/Kanal stimmt.

---

3 Vollständige Inventarliste (alle PHP-Dateien)

Datei	Klasse / Inhalt
src/ItkRobotsTxt.php	Plugin-Bootstrap (**extends Plugin**), i. d. R. ohne eigene **install**/**update**-Logik

Datei	Klasse / Inhalt
src/Storefront/Controller/RobotsTxtController.php	**RobotsTxtController** – einzige Request-Logik

---

4 Controller im Detail

Klasse: `ItkComputerGmbh\\ItkRobotsTxt\\Storefront\\Controller\\RobotsTxtController`
Basisklasse: `Shopware\\Storefront\\Controller\\StorefrontController` (Zugriff auf Container/Twig, hier aber ohne Template-Render).

Route (Attribut auf der Methode):

- **Pfad:** `/robots.txt`
- **Name:** `frontend.itk-robots-txt.page`
- **Methoden: GET**
- **Scope:** Klassen-Level **defaults: ['_routeScope' => ['storefront']]**

Ablauf robotsTxt(SalesChannelContext $context):**

1. **systemConfigService->get('ItkRobotsTxt.config.robotsTxtContent', $context->getSalesChannelId())** als String casten.
2. Wenn der String **nicht leer** ist → genau dieser Inhalt als Body.
3. Wenn **leer** → statischer Fallback **"User-agent: *\nAllow: /"**.
4. **Response** mit Status 200, Header:
    - **Content-Type: text/plain; charset=UTF-8**
    - **Cache-Control: public, max-age=3600**

Es gibt **keine** Template-Renderung, **keine** Event-Dispatcher-Calls, **keine** Validierung des Inhalts. Gewollt total stupide.

---

5 Konfiguration (`config.xml`)

Bitte bei "Alle Saleschannels" nichts konfigurieren. **Immer** einen Saleschannel oben auswählen.

Ein Eingabefeld:

- **Name: robotsTxtContent**
- **Typ: textarea**

Der technische Key in der Datenbank folgt dem Muster `ItkRobotsTxt.config.robotsTxtContent`.

---

6 Services (`services.xml`)

```xml
<service id="ItkComputerGmbh\ItkRobotsTxt\Storefront\Controller\
    RobotsTxtController" public="true">
    <argument type="service" id="Shopware\Core\System\SystemConfig\
    SystemConfigService"/>
    <call method="setContainer">…</call>
    <call method="setTwig">…</call>
    <tag name="controller.service_arguments"/>
</service>
```

- **public=“true”** – der Controller muss vom HTTP-Kernel instanziierbar sein.
- `controller.service_arguments` – Shopware/Symfony injiziert Request/Context in Controller-Methoden nach ihrer Konvention.
- **setContainer** / **setTwig** – Standard für Storefront-Controller, auch wenn diese Methode kein Twig nutzt. Technische Shopware Vorgabe.

Es gibt **keine** explizite Routen-Priorität in dieser XML-Datei. Welche Route /robots.txt „gewinnt“, hängt von der **Route-Registrierung** aller Bundles/Plugins ab. Wenn parallel noch eine Core- oder Theme-Route existiert, musst du bei Konflikten die **Routenliste** (**bin/console debug:router | findstr robots**) prüfen und ggf. Prioritäten im Attribut oder Bundle-Reihenfolge anpassen.

---

7 Routing (`routes.xml`)

Import des Controller-Verzeichnisses mit **type=“attribute”**, damit die **#[Route]**-Attribute ausgewertet werden.

---

8 Tests und lokales Nachvollziehen

1. Plugin aktivieren, im Admin Text setzen oder leer lassen.
2. Im Browser aufrufen: `https://<dein-host>/robots.txt`
3. Mit anderem Sales Channel (andere Domain) wiederholen und Inhalt vergleichen.
4. Optional: **bin/console debug:router frontend.itk-robots-txt.page** (Name exakt wie oben) zur Verifikation.

---

9 Debugging-Checkliste

1. **Route:** `debug:router` – wird `frontend.itk-robots-txt.page` registriert?
2. **Sales Channel:** Entspricht die aufgerufene Domain dem Kanal, für den du konfiguriert hast?
3. **Leer vs. null:** Der Code prüft **!== ''** nach String-Cast – nur echter Leerstring löst den Fallback aus.
4. **Cache:** Response-Header **Cache-Control** beachten; Proxys loggen.