

自测文档（Test Plan）

Demo: Edge Saved – Article Bookmarking（Edge SSR）

文档版本: v0.1

作者:（示例）Yan Huang

最后更新: 2025-12-24

适用范围: Edge Saved Web Application（Cloudflare Workers + React Router v7）

A screenshot of a terminal window with a dark background. The terminal shows a series of commands and their outputs. The first command is 'node -v' which outputs 'v20.19.6'. The second command is 'npm -v' which outputs '10.8.2'. The third command is 'wrangler -v' which outputs 'wrangler 4.54.0'. The terminal prompt is 'yanhuang@Yans-MacBook-Pro edge-saved %'.

1. 背景与目标

本文档描述 Edge Saved 项目的自测方案与验证结果，用于在提交代码前验证核心功能在 Edge SSR 场景下的正确性、稳定性与安全性。

测试重点包括：

- 用户收藏状态在 SSR 首屏渲染阶段的正确性
- Signed Cookie 的防篡改能力
- 收藏 / 取消收藏流程的幂等性
- Edge 环境下的基础性能表现

2. 被测功能概述

2.1 功能描述

Edge Saved 提供文章列表与收藏功能，允许用户在无需登录、无需数据库的情况下，通过 Signed Cookie 在浏览器侧持久化收藏状态，并在 Edge SSR 阶段完成首屏渲染。

核心功能包括：

- 展示文章列表及其收藏状态
- 支持用户对文章进行收藏与取消收藏操作
- 在页面刷新或重新访问时，保持收藏状态一致

- 在 Edge Runtime 中安全处理用户状态，防止客户端篡改

该功能主要用于验证在 Edge / SSR 场景下，如何在不引入后端状态存储系统的情况下，实现用户级状态的正确渲染与安全持久化。

2.2 功能边界与约束

为明确测试范围，本次自测对功能边界作如下约束说明：

- 不涉及用户登录、鉴权或账号体系
- 不涉及数据库、KV、对象存储等持久化系统
- 收藏状态仅在当前浏览器内有效，不支持跨设备同步
- 文章数据为静态示例数据，不涉及内容增删改
- 不考虑高并发写入或复杂业务规则

2.3 输入与输出说明

输入

- HTTP GET 请求：
 - `/`：访问文章列表页
 - `/saved`：访问已收藏文章列表页
- HTTP POST 请求：
 - 收藏 / 取消收藏操作（通过 React Router action 提交）
- Cookie：
 - `bookmarks`：包含用户收藏状态的 Signed Cookie

输出

- SSR HTML 页面：
 - 页面中包含文章列表及对应的收藏状态
- HTTP 响应头：
 - 在收藏状态发生变化时返回 `Set-Cookie`
- 页面行为：
 - 页面刷新后收藏状态保持一致
 - Cookie 异常时自动回退至默认状态

3. 功能测试用例

3.1 FT-01：首次访问（无 Cookie）

测试目的

验证在无任何历史状态的情况下，系统能够在 SSR 阶段正确渲染默认收藏状态。

前置条件

- 浏览器处于无痕窗口，或已清空 **bookmarks** Cookie

测试步骤

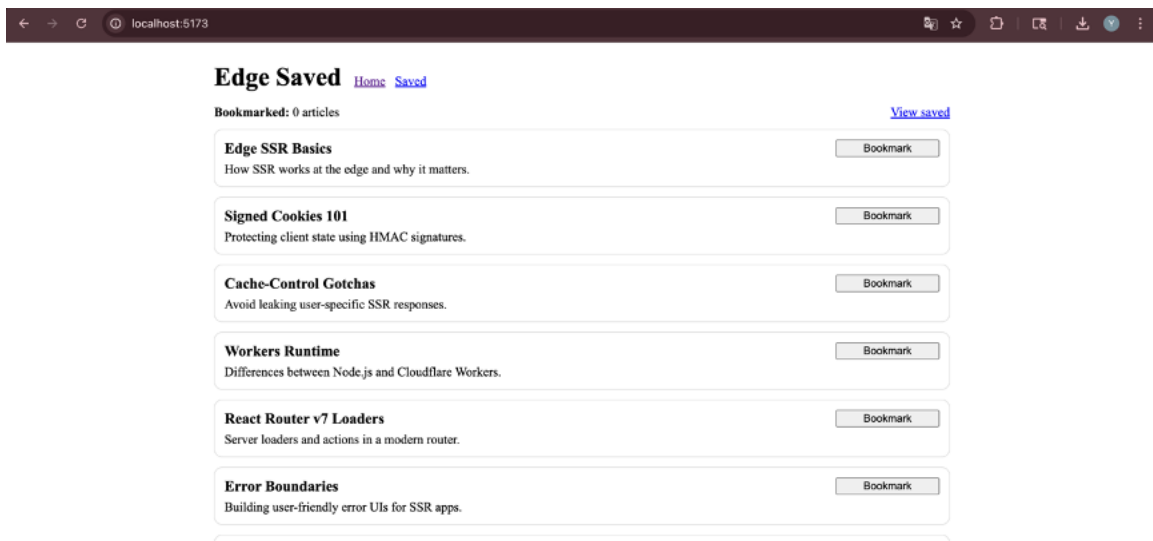
- 访问首页 **GET /**
- 不进行任何用户操作

预期结果

- 页面正常渲染
- 收藏数量显示为 **0 articles**
- 页面无报错或闪烁

实际结果

- 页面首屏显示 **Bookmarked: 0 articles**
- 所有文章均显示为未收藏状态



结论

✅ 通过

说明

该场景通过无 Cookie 状态访问首页进行验证，证明系统在 SSR 阶段能够正确处理“无状态”输入。

3.2 FT-02: 收藏文章

测试目的

验证用户点击 Bookmark 后，收藏状态能够被正确持久化。

前置条件

- 当前页面无收藏或部分收藏状态

测试步骤

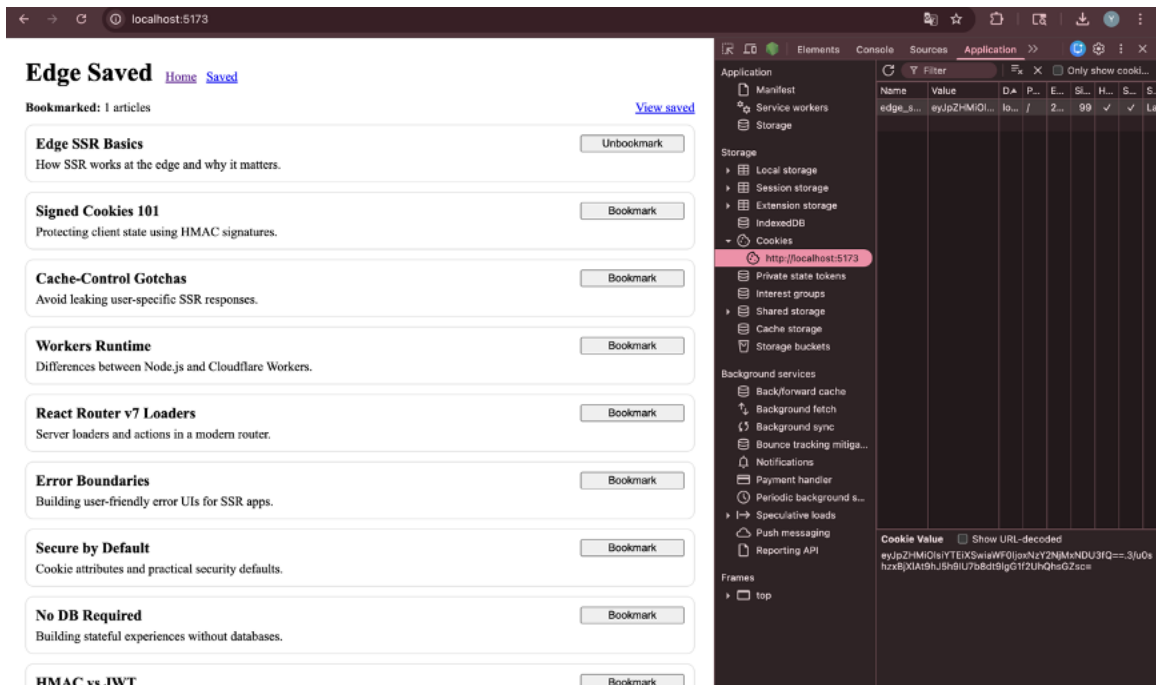
1. 在首页点击第一篇文章的 Bookmark 按钮
2. 等待页面刷新完成

预期结果

- 页面显示 Bookmarked: 1 articles
- 对应文章按钮状态变为 Unbookmark
- 浏览器中生成 bookmarks Cookie

实际结果

- 页面显示 Bookmarked: 1 articles
- 第一篇文章按钮变为 Unbookmark
- DevTools → Application → Cookies 中可见 bookmarks Cookie（见截图）



结论
✅ 通过

说明
截图中明确展示了收藏状态变化以及 Cookie 的生成，验证了服务端 action 正确更新用户状态。

3.3 FT-03：刷新页面保持收藏状态（SSR）

测试目的
验证收藏状态在页面刷新后仍能在 SSR 首屏阶段正确渲染。

前置条件

- 已存在有效的 **bookmarks** Cookie

测试步骤

1. 刷新浏览器页面（Cmd/Ctrl + R）

预期结果

- 页面首屏直接显示正确的收藏数量
- 不出现“先 0 后 1”的状态闪烁

实际结果

- 刷新后页面首屏直接显示 **Bookmarked: 1 articles**
- 收藏状态与刷新前一致

结论

 通过

说明

该结果表明收藏状态来源于 SSR loader 阶段，而非客户端计算。

3.4 FT-04: 取消收藏 (Unbookmark)

测试目的

验证取消收藏操作能够正确更新用户状态。

前置条件

- 至少存在一篇已收藏文章

测试步骤

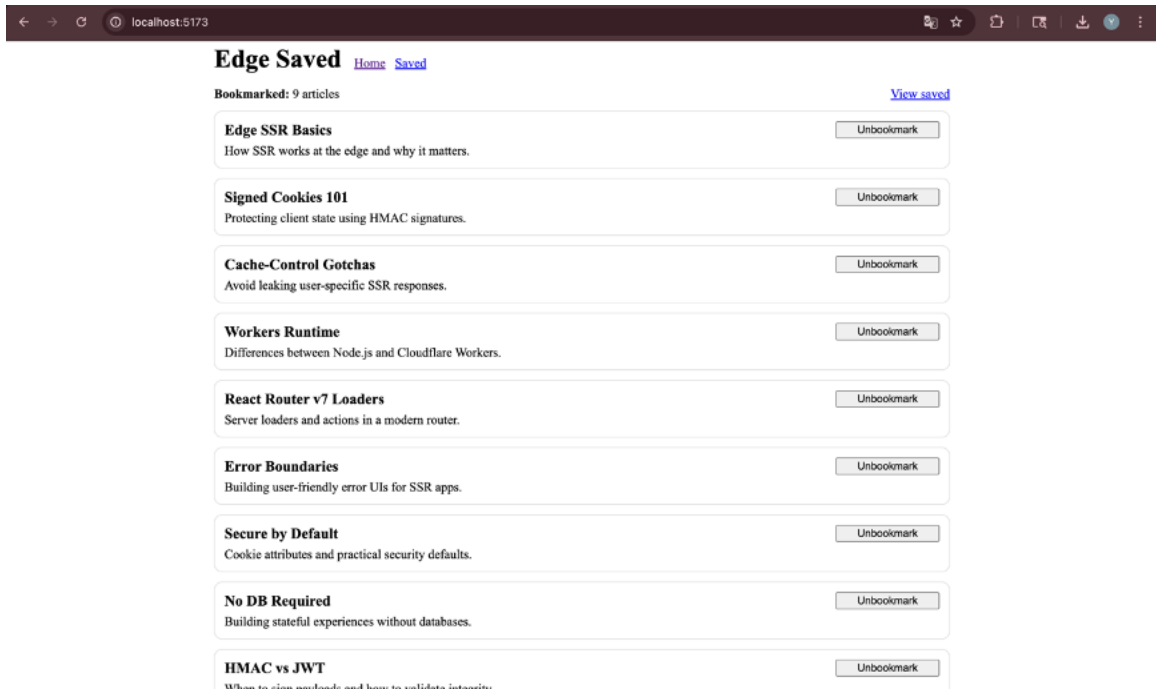
1. 点击已收藏文章的 **Unbookmark** 按钮
2. 等待页面刷新

预期结果

- 收藏数量减少
- 对应文章恢复为未收藏状态

实际结果

- 页面显示收藏数量减少
- 已取消收藏的文章按钮恢复为 **Bookmark**



结论

✅ 通过

说明

通过实际操作验证取消收藏流程，服务端正确更新 Cookie 状态。

3.5 FT-05: 查看收藏页面（/saved）

测试目的

验证收藏页面仅展示用户已收藏的文章。

前置条件

- 用户存在至少一篇已收藏文章

测试步骤

1. 点击页面中的 **Saved** 或 **View saved**
2. 进入 **/saved** 页面

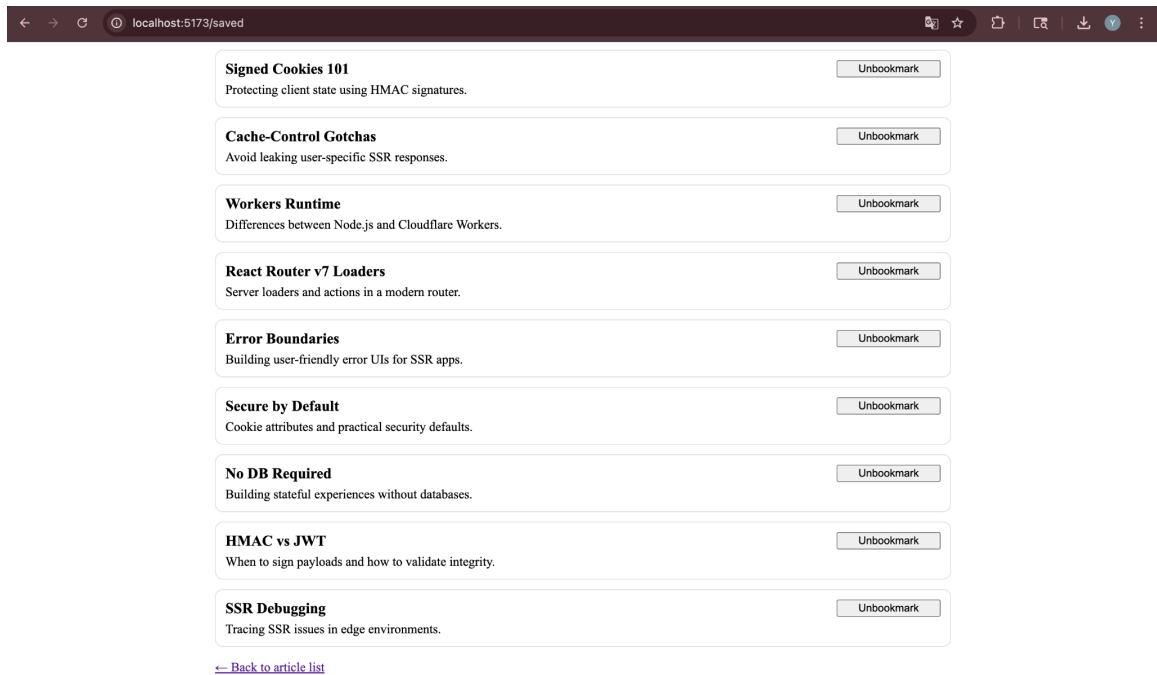
预期结果

- 页面仅显示已收藏文章

- 未收藏文章不显示

实际结果

- 收藏页面仅展示已收藏文章列表
- 页面渲染正确，无异常内容



结论

✅ 通过

3.6 FT-06: Cookie 被篡改（安全性验证）

测试目的

验证 Signed Cookie 能防止客户端篡改用户状态。

前置条件

- 浏览器中存在 **bookmarks** Cookie

测试步骤

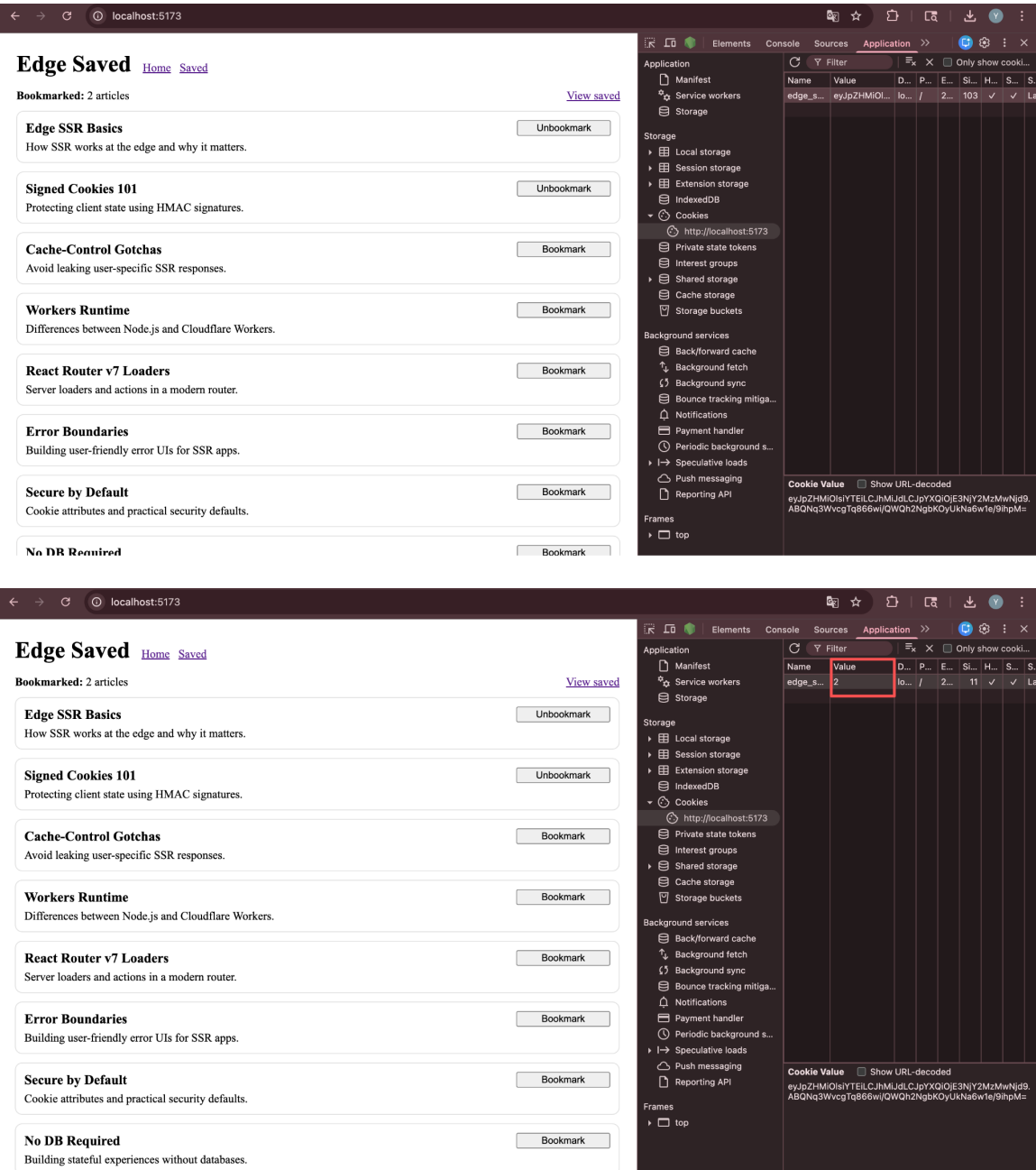
1. 打开 DevTools → Application → Cookies
2. 手动修改 **bookmarks** Cookie 的值
3. 刷新页面

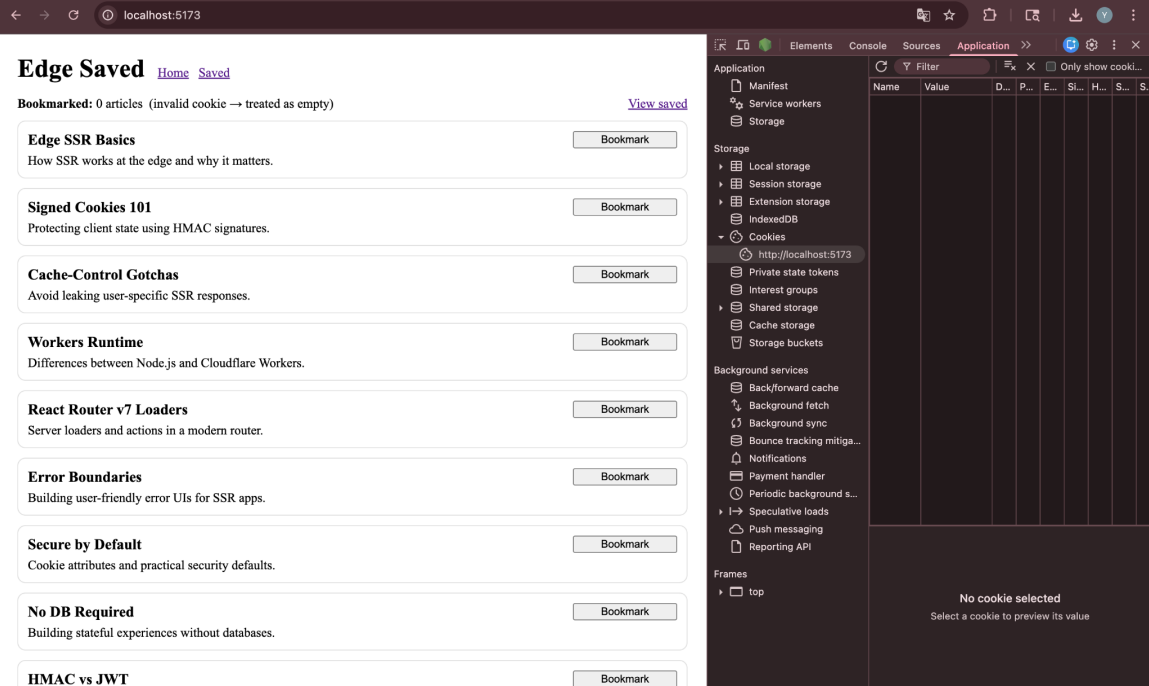
预期结果

- 服务端忽略被篡改的 Cookie
- 收藏状态回退为默认状态

实际结果

- 页面显示 **Bookmarked: 0 articles**
- 被篡改的 Cookie 未生效





结论

✅ 通过

说明

该测试验证了 Cookie 签名机制能够有效防止客户端伪造收藏状态。