ShortLinks user guide

What you get

ShortLinks provides a lightweight URL shortener for ASP.NET Core that supports:

- Create short link → returns code and shortUrl
- Resolve/Redirect short link (/s/{code}) → redirects to original URL
- Expiring links (ExpireAtUtc)
- One-time / limited-use links (OneTime, MaxUses) — if enabled in your version
- Storage providers:
    - EF Core (SQL Server)
    - Redis

1) Packages

Install what you need:

Core (required)

- ShortLinks.Core
- ShortLinks.Abstractions (will be installed automatically via dependencies)

Storage (choose ONE)

- ShortLinks.Storage.EFCore (SQL Server / EF Core)
- ShortLinks.Storage.Redis (Redis)

2) Consumer setup (ASP.NET Core Web API with Controllers)

2.1 Create a new Web API project (optional)

2.2 Install packages (one of the following):

- EF Core (SQL Server)
  1. dotnet add package ShortLinks.Core
  2. dotnet add package ShortLinks.Storage.EFCore
  3. dotnet add package Microsoft.EntityFrameworkCore.SqlServer

- Redis
  1. dotnet add package ShortLinks.Core
  2. dotnet add package ShortLinks.Storage.Redis
  3. dotnet add package StackExchange.Redis

3) Configure Program.cs

3.1 Register ShortLinks Core (required)

Add this in Program.cs:

```
builder.Services.AddShortLinksCore(opt =>
{
  opt.BaseUrl = "https://localhost:5001/s"; // returned in Create result
  opt.DefaultCodeLength = 8;
  opt.MaxCreateAttempts = 15;
  opt.TrackHits = true;
});
```

BaseUrl should match your real domain in production: https://your-domain.com/s

3.2 Choose a storage provider

Option A — EF Core / SQL Server

appsettings.json:

```
{
  "ConnectionStrings": {
    "ShortLinks":
"Server=.;Database=ShortLinksDb;Trusted_Connection=True;TrustServerCertificate=True"
  }
}
```

Program.cs:

```
using Microsoft.EntityFrameworkCore;
using ShortLinks.Storage.EFCore;

builder.Services.AddShortLinksEfCore(options =>
{
options.UseSqlServer(builder.Configuration.GetConnectionString("ShortLinksDb"));
});
```

Apply migrations (recommended for dev):

```
using (var scope = app.Services.CreateScope())
{
   var db =    scope.ServiceProvider.GetRequiredService<ShortLinksDbContext>();
   db.Database.Migrate();
}
```

Option B — Redis

appsettings.json:

```
{
  "Redis": {
    "ConnectionString": "localhost:6379,abortConnect=false"
  }
}
```

Program.cs

```
using ShortLinks.Storage.Redis;

builder.Services.AddShortLinksRedis(builder.Configuration["Redis:ConnectionStri"opt =>
{
    opt.KeyPrefix = "shortlinks:";
});
```

4) Controllers (required)

4.1 Create controller (Create short links)

Create Controllers/ShortLinksController.cs:

```
using Microsoft.AspNetCore.Mvc;
using ShortLinks.Abstractions;
namespace YourApp.Controllers;

[ApiController]
[Route("api/shortlinks")]
public class ShortLinksController : ControllerBase
{
  private readonly IShortLinkService _service;
  public ShortLinksController(IShortLinkService service)
  {
    _service = service;
  }

  [HTTP Post]
  public async Task<IActionResult> Create([FromBody]
CreateShortLinkRequest request, CancellationToken ct)
  {
    var result = await _service.CreateAsync(request, ct);
    return Ok(result);
  }
}
```

DTO Reference (Exact)

CreateShortLinkRequest:

```
public sealed record CreateShortLinkRequest(
  string OriginalUrl,
  DateTime? ExpireAtUtc = null,
  string? CustomCode = null,
  bool OneTime = false,
  int? MaxUses = null
);
```

CreateShortLinkResult:

```
public sealed record CreateShortLinkResult(
  string Code,
  string OriginalUrl,
  string ShortUrl
);
```

ResolveShortLinkResult:

```
public sealed record ResolveShortLinkResult(
  string Code,
  string OriginalUrl
);
```

4.2 Redirect controller (/s/{code})

Create Controllers/RedirectController.cs:

```
using Microsoft.AspNetCore.Mvc;
using ShortLinks.Abstractions;

namespace YourApp.Controllers;
[ApiController]
public class RedirectController : ControllerBase
{
  private readonly IShortLinkService _service;

  public RedirectController(IShortLinkService service)
  {
    _service = service;
  }
  [HttpGet("s/{code}")]
  public async Task<IActionResult> Go(string code, CancellationToken ct)
  {
    var resolved = await _service.ResolveAsync(code, ct);
    if (resolved is null)
      return NotFound();
    return Redirect(resolved.OriginalUrl);
  }
}
```

5) How to use the API

5.1 Create a normal short link:

POST /api/shortlinks

```
{
  "originalUrl": "https://example.com/pay?invoiceId=123"
}
```

Response:

```
{
  "code": "Ab12Cd34",
  "shortUrl": "https://localhost:5001/s/Ab12Cd34",
  "originalUrl": "https://example.com/pay?invoiceId=123"
}
```

Open:
GET https://localhost:5001/s/Ab12Cd34 → redirects.

6) Expiring + One-time / Limited-use links (if your version includes it)

6.1 Expiring link:

```
{
  "originalUrl": "https://example.com/reset?token=abc",
  "expireAtUtc": "2026-02-01T12:00:00Z"
}
```

6.2 One-time link:

```
{
  "originalUrl": "https://example.com/reset?token=abc",
  "oneTime": true
}
```

6.3 Limited-use link (e.g., 5 uses):

```
{
  "originalUrl": "https://example.com/download?id=7",
  "maxUses": 5
}
```

Expected behavior:

- After limit reached, GET /s/{code} → 404 Not Found (or whatever you implement).

7) Running Redis locally (Docker)

```
docker run -d --name redis-shortlinks -p 6379:6379 redis
```

Connection string:

```
localhost:6379,abortConnect=false
```

Check stored keys:

```
docker exec -it redis-shortlinks redis-cli
KEYS shortlinks:*
GET shortlinks:YOUR_CODE
TTL shortlinks:YOUR_CODE
```

8) Troubleshooting:

“UseSqlServer does not exist”

Install SQL Server provider and add the using:

- Microsoft.EntityFrameworkCore.SqlServer
- using Microsoft.EntityFrameworkCore;

“AddShortLinksRedis does not exist”

- Ensure API references ShortLinks.Storage.Redis
- Add using ShortLinks.Storage.Redis;
- Ensure extension class is public static

Redis “cannot connect”

- Confirm container port mapping
- Use correct host:
    - App running locally → localhost:PORT
    - App running in Docker → redis-container-name:6379