PREPÁRATE
PARA SER EL
MEJOR

+ ENTREMIENTO
EXPERIENCIA

BIENVENIDOS.

ENTRENAMIENTO
EXPERIENCIA

GALAXY
TRAINING

APLICACIONES WEB CON ASP NET

Ing. Erick Arostegui Cunza
Instructor

earostegui@galaxy.edu.pe

AGENDA

Componentes MVC Y aplicando seguridad al Website

- Aplicando inyección de dependencias.
- Usando helpers personalizados.
- Usando filtros personalizados (action filters).
- Gestión de excepciones.
- Gestión de sesiones, cookies.
- Gestión de páginas en Caché (local, otro servidor, Redis).
- Gestión de la autenticación.
- Autorización personalizada usando Action Filters.

Inyección de dependencias (DI).

Intermedio

Un conjunto de principios y patrones de diseño de software que nos permiten desarrollar código débilmente acoplado.

Van Deursen and Seeman. Dependency Injection in .NET. Manning, 2018.

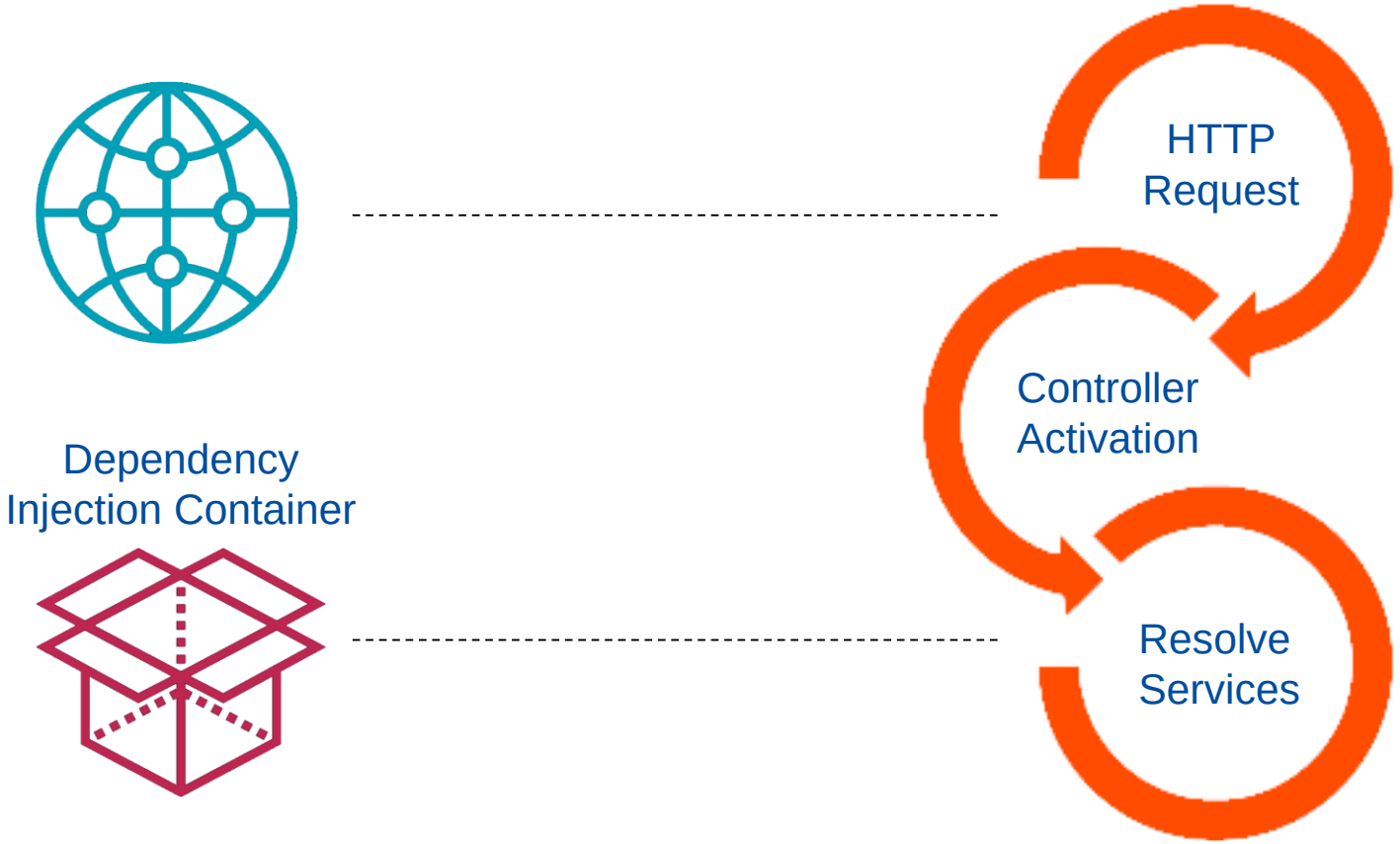
Inyección nativa en Net Core.

Intermedio

ASP.NET Core entrega su propio contenedor de inyección de dependencia. Este contenedor ya tiene por defecto todos los servicios necesarios para la ejecución de los procesos.

Inyección nativa en Net Core.

Intermedio



Tiempos de vida del servicio

Intermedio

Transient

Creado cada vez que es solicitado

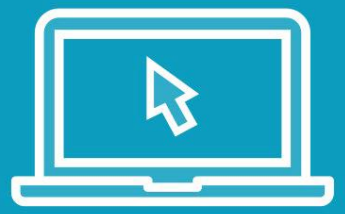
Singleton

Creado una vez durante la vida útil de la aplicación.

Scoped

Creado una vez durante la vida del request

Demo



Inyección de dependencias (DI).

¿Por qué construir tu propio Helper de MVC?

- Simplifica tu codificación de Razor.
- Puede cambiar el HTML emitido en un solo lugar.
- Envuelva varias líneas de HTML en un Helper de HTML.
- Añadir atributos adicionales.
- Añadir nuevas tecnologías a medida que estén disponibles.
- Proporcionar varias sobrecargas para cada Helper.

Los Helpers de MVC son limitados

- Los objetos anónimos se usan con demasiada frecuencia.
- Los objetos anónimos pueden conducir a errores de tiempo de ejecución.
- No existen Helpers para todos los controles.

Creando Helpers

Usando StringBuilder

- Paso 1: construir una clase estática
- Paso 2: agrega un método estático para devolver un MvcHtmlString
- Paso 3: Usa la clase StringBuilder para construir HTML
- Paso 4: Devuelva el HTML como un MvcHtmlString

Usando TagBuilder

- TagBuilder está diseñado específicamente para crear etiquetas HTML
- Añade elementos de apertura y cierre
- Agrega atributos a los elementos.
- Métodos para generar atributos válidos de 'nombre' e 'id'

Usando helpers personalizados

Intermedio

Los Tag Helpers permiten que el código C # del lado del servidor participe en la creación y renderización de elementos HTML en archivos Razor

Tag Helpers

Son similares a los HTML Helpers

HTML-friendly

Soporte IntelliSense

Predefinidos o personalizados

JavaScript Tag Helpers

asp-src-include

asp-src-exclude

asp-fallback-src

asp-fallback-test

CSS Tag Helpers

asp-href-include

asp-href-exclude

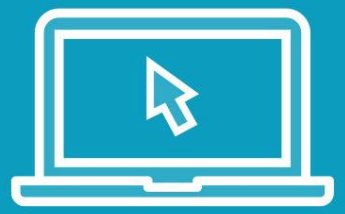
asp-fallback-href-include

asp-fallback-href-test-class asp-fallback-href-test-property asp-fallback-href-test-value

Inyección de dependencias (DI).

Intermedio

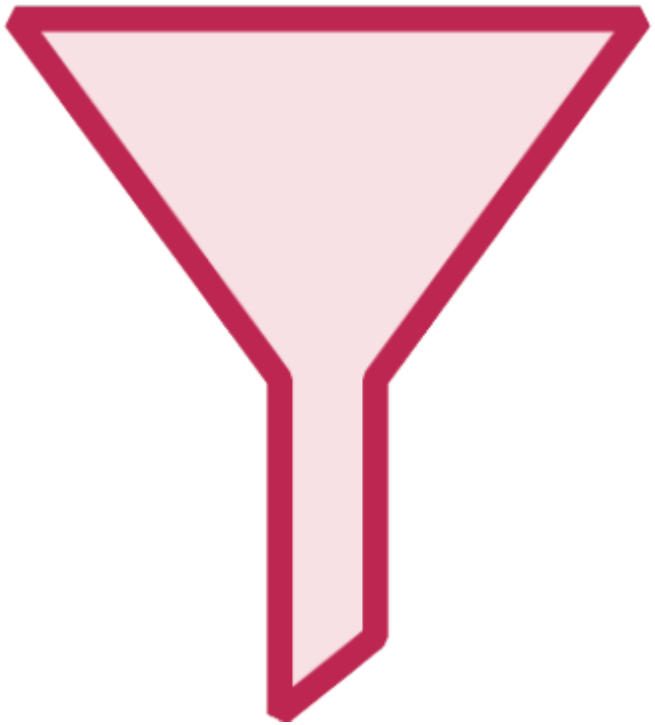
Demo



Helpers

Usando filtros personalizados (action filters).

Intermedio



Nos permite agregar lógica a una solicitud de MVCs

- Antes o después

Se utiliza a menudo funciones transversales

- Evitar la duplicación de código

Usos comunes

- Autorización
- Requerir HTTPS

Tipos de filtros

- Authorization
- Resource
- Action
- Exception
- Result

Ambito del filtro

Action Method

Ejecutado solo anivel del Action

Controller

Ejecutado para todos los Action’s que pertenecen a la clase

Global

Ejecutado para todos los Action’s presentes en la aplicación

Usando filtros personalizados (action filters).

Intermedio

Authorization Filters

- Primero en ser ejecutado
- Solo tiene el método Before
- [Authorize]

Usando filtros personalizados (action filters).

Intermedio

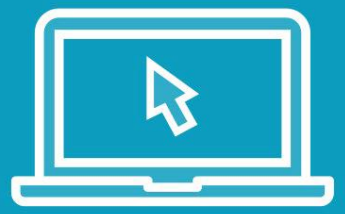
Action Filters

- Propósito general
- Puede interrumpir el flujo antes de la ejecución del action.
- Puede cambiar el resultado del action
- Se implementa a través de la interfaz IActionFilter

Usando filtros personalizados (action filters).

Intermedio

Demo



Creando un Filtro de autorización

Creando un Filtro de Action

Usando filtros personalizados (action filters).

Intermedio

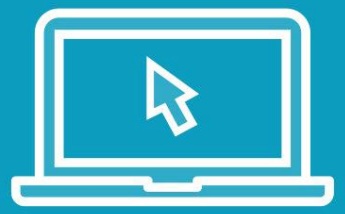
Los filtros globales se aplican a todos los Action’s

Cualquier filtro puede ser aplicado

Usando filtros personalizados (action filters).

Intermedio

Demo

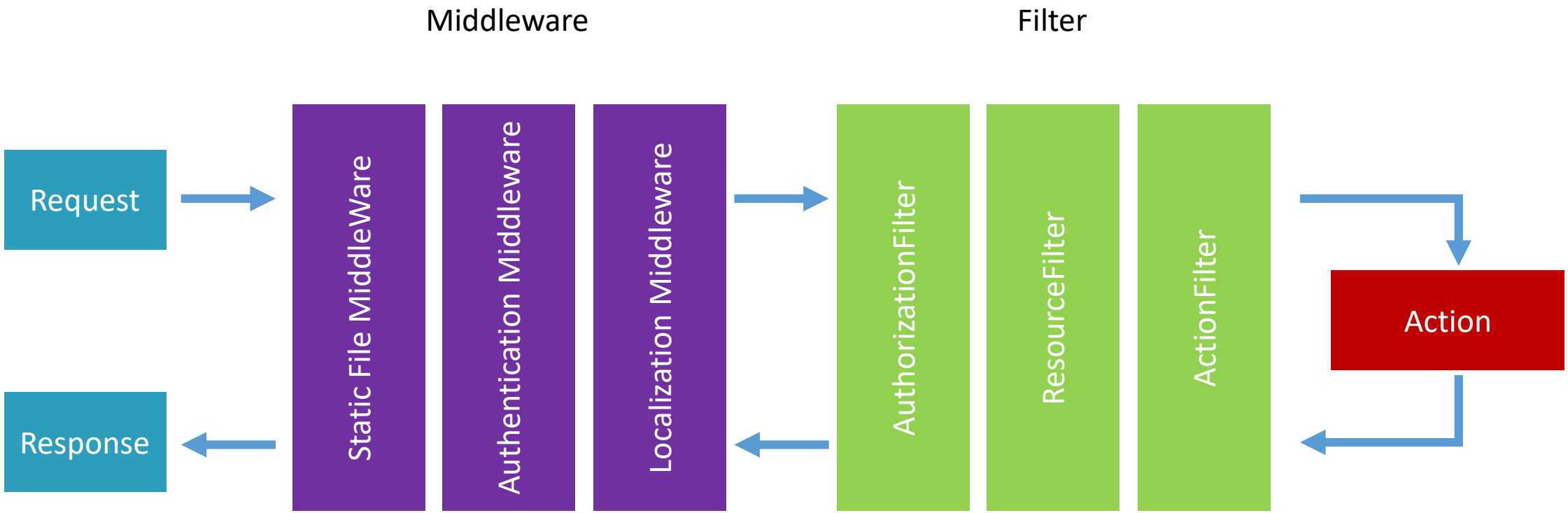


Trabajando con filtros globales

Middleware

Middleware es un software que se ensambla en un canal (pipe) de aplicaciones para manejar solicitudes y respuestas

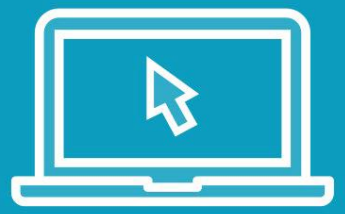
Middleware


Middleware
Filter
Request
Response
Static File MiddleWare
Authentication Middleware
Localization Middleware
AuthorizationFilter
ResourceFilter
ActionFilter
Action

Gestión de excepciones.

Intermedio

Demo



Creando un filtro de excepción

Creando un middleware de excepción

Estado (State)

Un programa se describe a través de estados, está diseñado para recordar eventos anteriores o interacciones de usuario; la información recordada se llama el estado del sistema

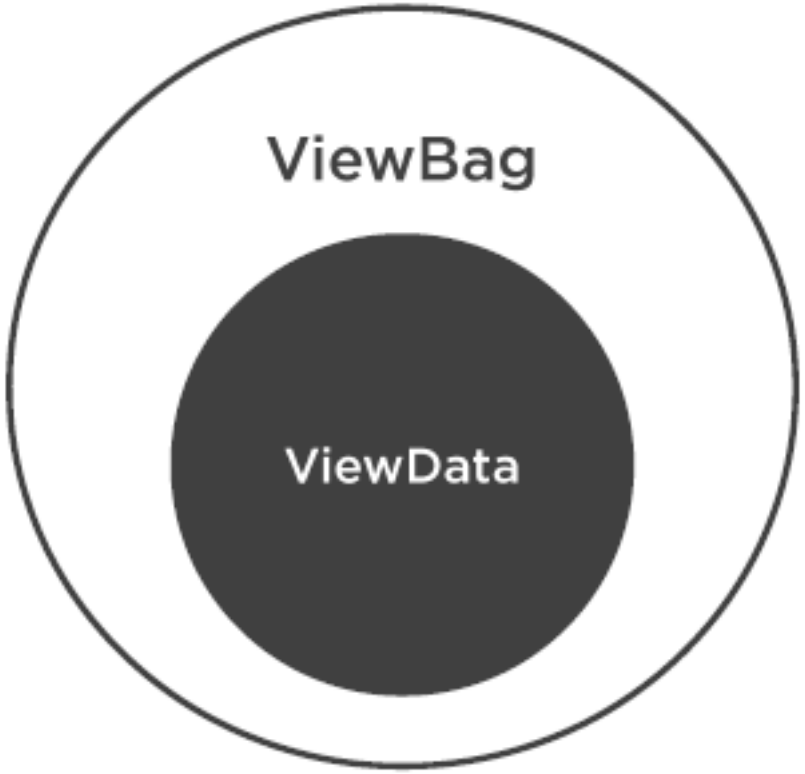
¿Porqué evitar los estados?

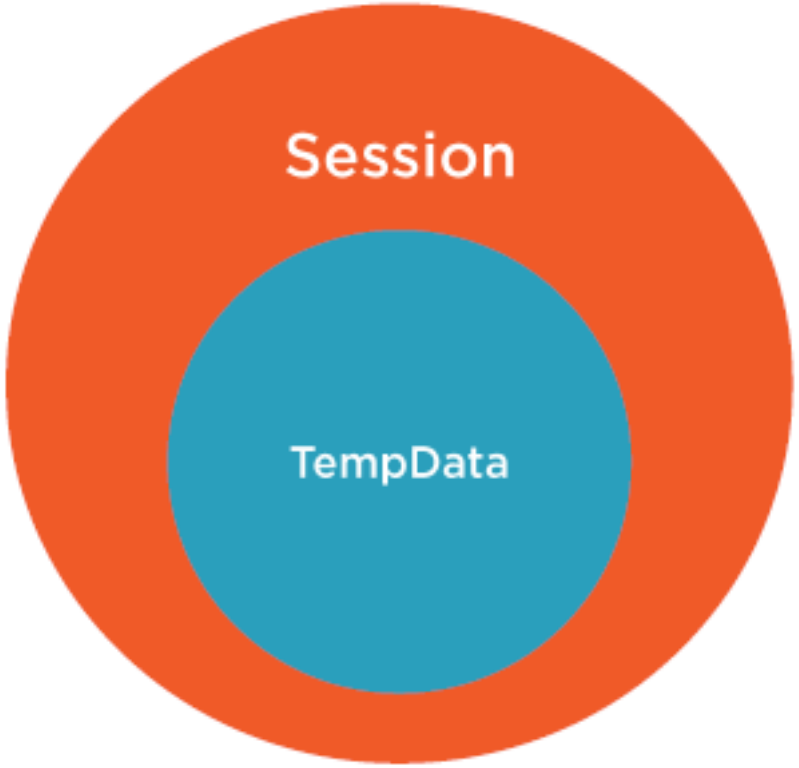
Las aplicaciones con estado son difíciles de escalar a comparación de las que no manejan estados.

El estado introduce complejidad

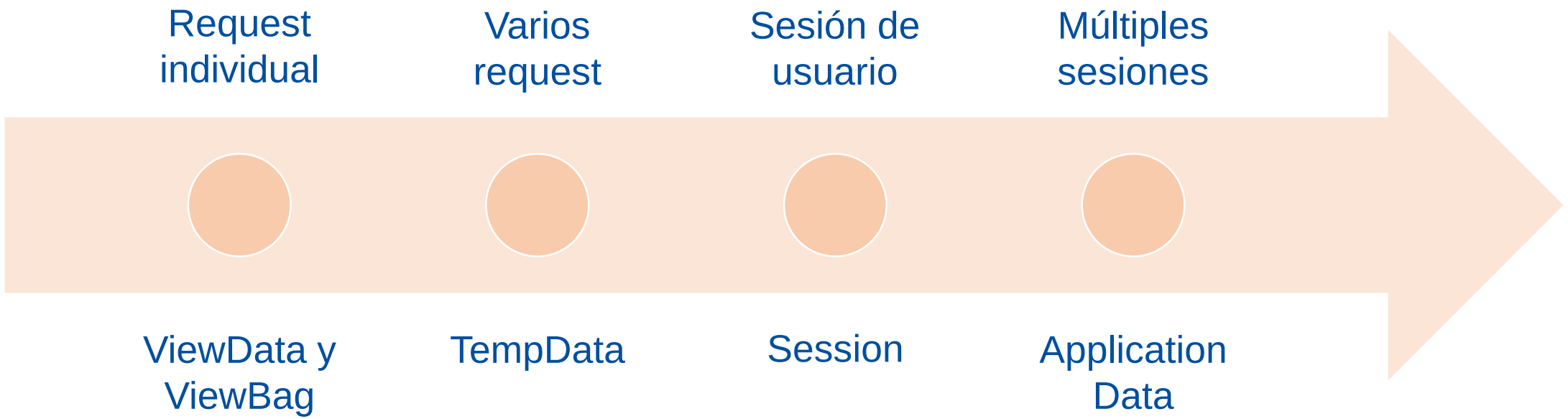
Algo siempre es más caro que nada.

Herramientas de manejo de estados para Razor





Volatilidad



¿Donde se almacenan las sesiones?

Se almacenan por defecto en la memoria.

Una granja de servidores necesitan persistencia de sesiones.

- Mejor es evitar.
- La experiencia de usuario puede verse afectada

Proveedores de sesión

Proveedores personalizados.

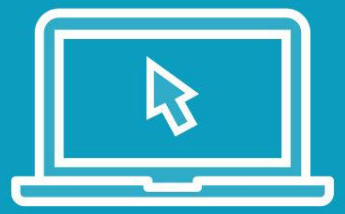
Soluciones existentes.

- Sql Server
- Redis Cache
- Memcached
- Ncached
- RavenDB

Gestión de sesiones, cookies.

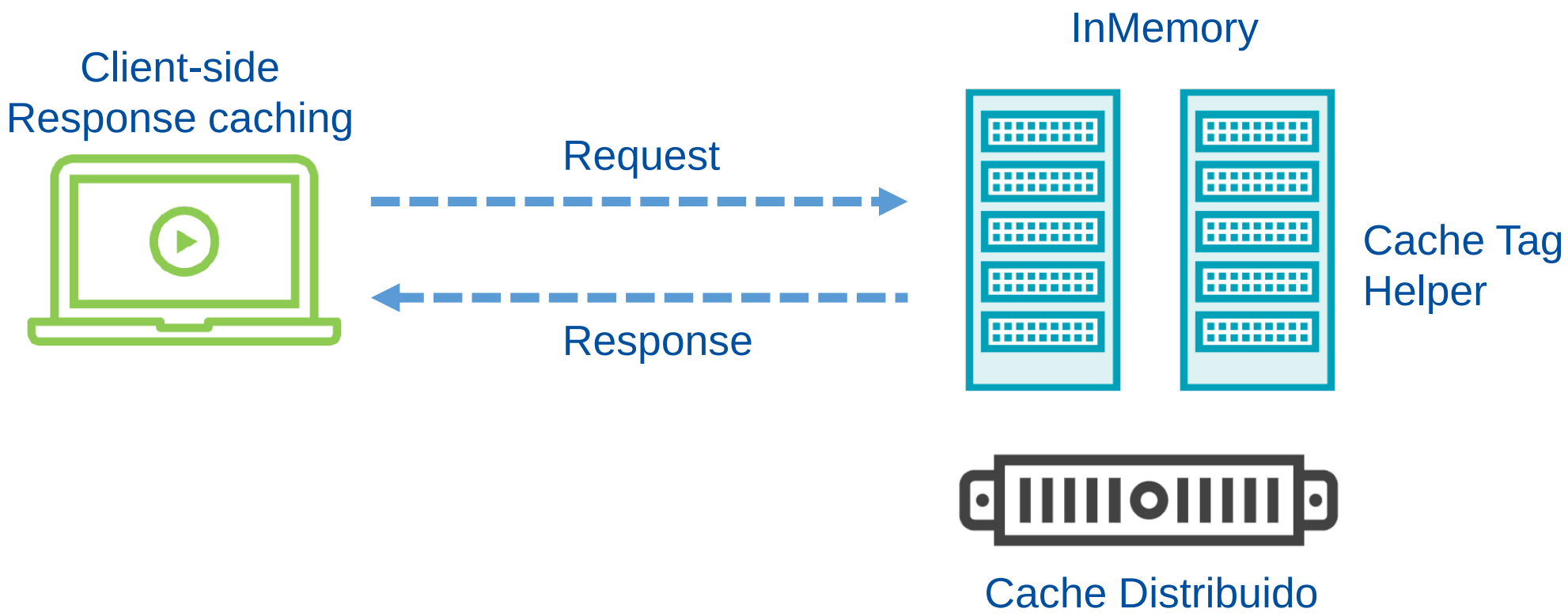
Intermedio

Demo



Sesiones

Opciones de Cache



Gestión de páginas en Caché (local, otro servidor, Redis).

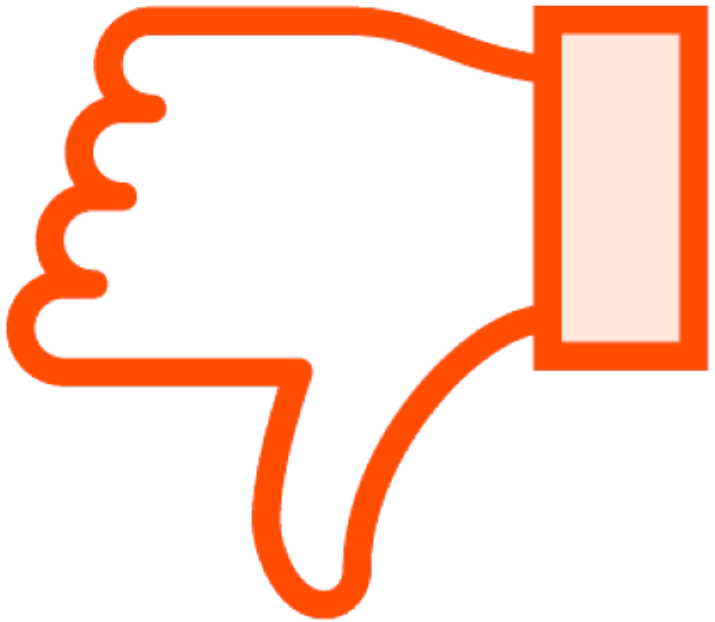
Intermedio

El almacenamiento en caché puede acelerar en gran medida un sitio web.

Normalmente se utiliza para datos que no sufrirán cambios por un periodo largo de tiempo

Gestión de páginas en Caché (local, otro servidor, Redis).

Intermedio



El uso del almacenamiento en caché requiere una planificación cuidadosa

De lo contrario, pueden aparecer extraños efectos secundario.

In-memory Caching

La forma mas simple

IMemoryCache

Trabajar con sesiones persistentes

Puede trabajar con cualquier tipo de objeto

MemoryCacheEntryOptions

Absolute expiration

Sliding expiration

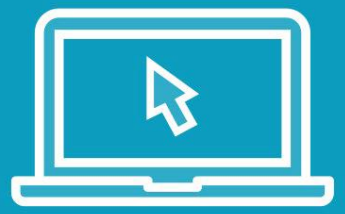
Cache priority

PostEvictionDelegate

Gestión de páginas en Caché (local, otro servidor, Redis).

Intermedio

Demo



Agregando soporte para IMemoryCache

Utilizando IMemoryCache

Cache Tag Helper

- Del lado del servidor
- Usa IMemoryCache
- Requiere persistencia de sesión

Opciones de expiración

expires-after

expires-on

expires-sliding

Opciones Vary-by

vary-by-user

vary-by-route

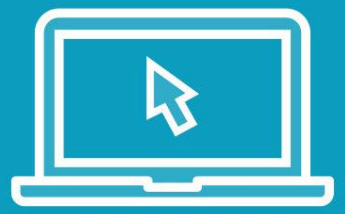
vary-by-query

vary-by-cookie

vary-by-header

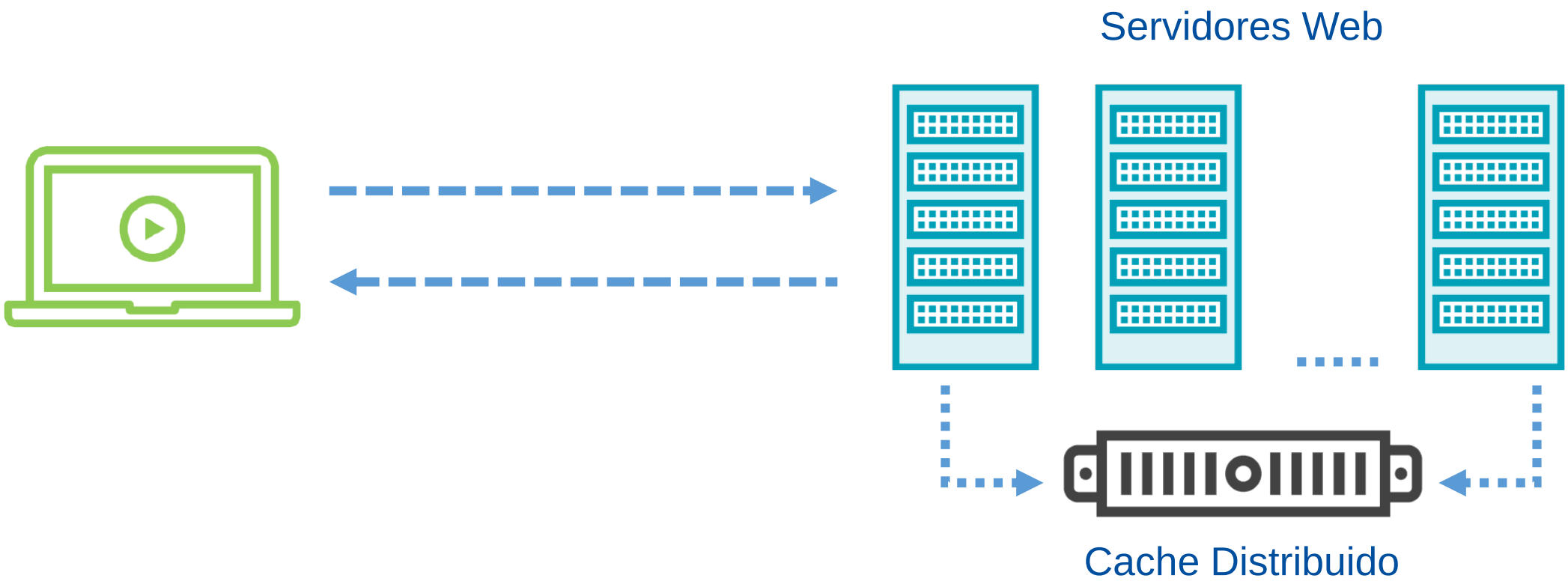
vary-by

Demo



Usando Cache Tag Helper

Cache distribuido



Gestión de páginas en Caché (local, otro servidor, Redis).

Intermedio

No se requiere sesiones adheridas

- Idéntico en todos los servidores

Escalable

No se ve impactado por los reinicios del servidor

Mejor rendimiento para el almacenamiento de datos.

Soporte Incorporado en ASP.NET Core

Redis

SQL Server

IDistributedCache

Get & GetAsync

Set & SetAsync

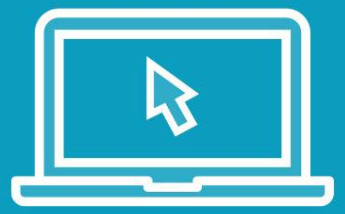
Refresh & RefreshAsync

Remove & RemoveAsync

Gestión de páginas en Caché (local, otro servidor, Redis).

Intermedio

Demo



Configurando Redis

Utilizando Cache distribuida

Response Caching

- Basado en encabezados
- Caching del lado del cliente
- Limita la carga en el servidor.
- Utiliza el Atributo ResponseCache

Opciones disponibles

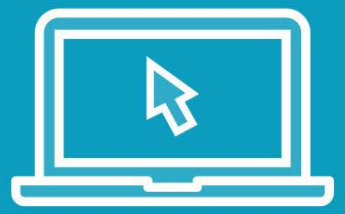
Location

Duration

NoStore

VaryByHeader

Demo

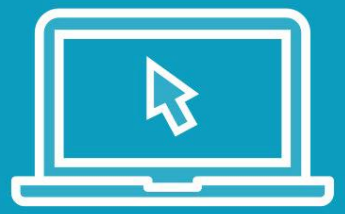


Utilizando Response caching

Gestión de la autenticación.

Intermedio

Demo



Local Login

Social Login

IdentityServer


GALAXY
TRAINING