**TempData in ASP.NET Core MVC**

TempData ist dafür gemacht, Daten *zwischen zwei HTTP-Requests* zu speichern – etwa von einer POST-Action zur GET-Action nach einem Redirect.

|  |  |
| --- | --- |
| Merkmal | Beschreibung |
| Typ | TempDataDictionary |
| Zugriff | TempData["key"] |
| Lebensdauer | Bis zum nächsten Request (danach gelöscht) |
| Einsatzgebiet | Übergeben von Daten zwischen zwei Actions (z. B. bei Redirect) |
| Typprüfung | Nein (wie bei ViewData) |
| Speicherung | Standardmäßig über Cookies oder Session (konfigurierbar) |
| Datenformate | Nur einfache Typen (z. B. string, int, bool, …) |

**Vergleich zu ViewData und ViewBag**

|  |  |  |  |
| --- | --- | --- | --- |
| Merkmal | ViewData | ViewBag | TempData |
| Gültigkeit | Aktueller Request | Aktueller Request | Über mehrere Requests hinweg |
| Typ | Dictionary | dynamic | Dictionary |
| Anwendungsfall | Controller → View | Controller → View | Redirect → View |
| Automatisch gelöscht? | – | – | Ja, nach Zugriff gelöscht |

**Beispiel**

Ein Mini-Formular, das beim Speichern an eine andere View weiterleitet und dort eine Erfolgsmeldung anzeigt.

**Projektstruktur**

* Controller: ProdukteController
* Views:
  + Produkte/Erfassen.cshtml – Formular
  + Produkte/Bestaetigung.cshtml – Anzeige von TempData

**Controller: ProdukteController.cs**

public class ProdukteController : Controller

{

[HttpGet]

public IActionResult Erfassen()

{

return View();

}

[HttpPost]

public IActionResult Erfassen(string produktname)

{

// Hier würde man normalerweise speichern

TempData["Nachricht"] = $"Das Produkt '{produktname}' wurde gespeichert.";

return RedirectToAction("Bestaetigung");

}

public IActionResult Bestaetigung()

{

return View();

}

}

**View: Erfassen.cshtml**

@{

ViewData["Title"] = "Produkt erfassen";

}

<h2>Produkt erfassen</h2>

<form asp-action="Erfassen" method="post">

<label>Produktname:</label>

<input type="text" name="produktname" />

<button type="submit">Speichern</button>

</form>

**View: Bestaetigung.cshtml**

@{

ViewData["Title"] = "Bestätigung";

}

@if (TempData["Nachricht"] != null)

{

<div class="alert alert-success">

@TempData["Nachricht"]

</div>

}

else

{

<p>Keine Nachricht vorhanden.</p>

}

**Übung**

Erstellen Sie eine neue Action Kontakt, in der Benutzer einen Namen und eine Nachricht absenden können.  
Leiten Sie anschließend zur View KontaktBestaetigung weiter, wo eine Dankesmeldung mit TempData angezeigt wird.

**Anforderung:**

* Name und Nachricht werden über POST übergeben
* TempData enthält eine Bestätigung wie:  
  „Danke Max für Ihre Nachricht!“

**Merke**

* TempData eignet sich zur Weitergabe von Daten über **Redirects** hinweg.
* Typischer Anwendungsfall: Erfolgsmeldung nach POST → RedirectToAction → Anzeige in View.
* Nach dem **Zugriff wird TempData gelöscht**, außer man verwendet Peek() oder Keep().

**TempData "bewusst" länger behalten**

**Hinweis: Keep() und Peek()**

* TempData.Keep("Nachricht") → verhindert das Löschen nach Zugriff
* TempData.Peek("Nachricht") → liest, aber löscht nicht

string meldung = TempData.Peek("Nachricht") as string;

**Lebensdauer von TempData nach Peek() oder Keep()**

**Ohne Peek() oder Keep()**:  
TempData wird **nach dem ersten Zugriff gelöscht** (egal ob in der View oder im Controller).

**Mit Peek() oder Keep()**:  
Die Daten bleiben **noch bis zum nächsten Request** erhalten – **danach werden sie trotzdem gelöscht**, **auch wenn sie nicht erneut gelesen wurden**.

**Zusammenfassung**

|  |  |
| --- | --- |
| Zugriffsmethode | Verhalten |
| TempData["key"] | Liest **und löscht** sofort nach dem Zugriff |
| TempData.Peek("key") | Liest **ohne zu löschen** – bleibt bis zum nächsten Request |
| TempData.Keep("key") | Verhindert das Löschen – bleibt bis zum nächsten Request |

**Achtung:** Auch mit Keep() oder Peek() bleibt der Eintrag **nicht dauerhaft** erhalten.  
Spätestens **nach dem nächsten Request** (egal ob Zugriff erfolgt oder nicht) ist die Information **weg**.

**Beispiel:**

public IActionResult ErsteSeite()

{

TempData["Hinweis"] = "Willkommen!";

return RedirectToAction("ZweiteSeite");

}

public IActionResult ZweiteSeite()

{

var meldung = TempData.Peek("Hinweis");

// TempData["Hinweis"] ist weiterhin vorhanden

return View();

}

// Dritter Aufruf (dritte Seite) => Hinweis ist gelöscht