DEVMASTER.VN

Microsoft .NET MVC 5

Bài 08

Quản lý State và tối ưu hóa

Mục tiêu

- Giới thiệu về quản lý trạng thái
- Cookies
- Tempdata
- Application State
- Session State
- Tối ưu hiệu suất ứng dụng web
- Caching
- Sử dụng Bundling và Minification

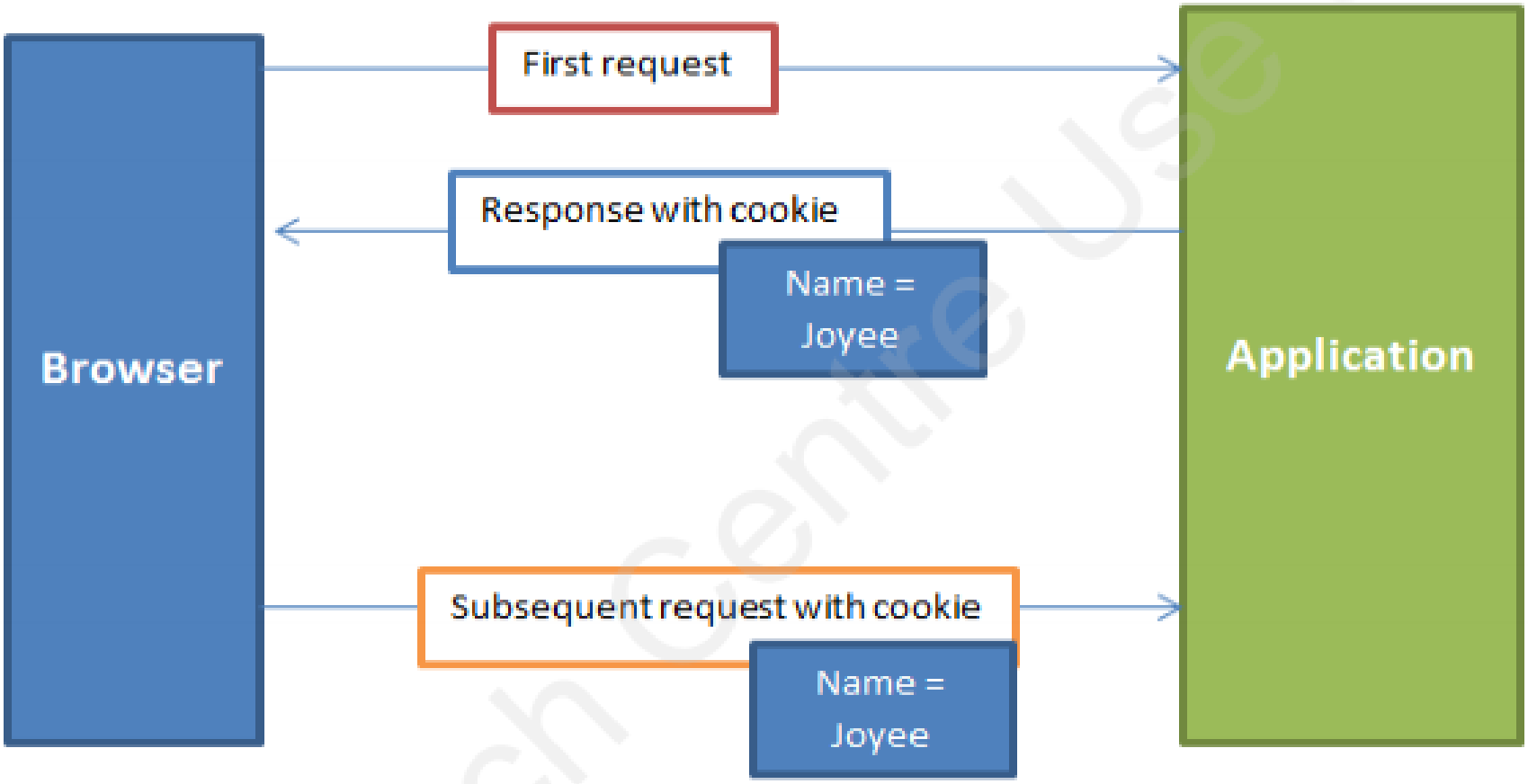
Quản lý trạng thái

- Các trang web sử dụng giao thức HTTP giao tiếp với Web server. HTTP là giao thức phi trang thái và không tự động nhận biết các request được gửi từ cùng hay từ các client khác nhau.
- ASP.NET MVC cung cấp một số đối tượng để quản lý trạng thái của Client như :
  - Cookies
  - TempData
  - Application State
  - Session State

Cookies

- Cookies là những phần thông tin nhỏ được lưu trữ ở máy client như IP address, loại trình duyệt, hệ điều hành, lần ghé thăm cuối cùng, thông tin đăng nhập.
- Cookies được gửi tới máy client theo trang trả về.
- Khi trình duyệt gửi request trên trang trong lần kế tiếp thì cookies sẽ được gửi đi kèm theo.
- Web server đọc cookie và lấy giá trị của chúng, sau đó nó sẽ xử lý tùy theo thông tin và trả về cho client.

Cookies


Browser
First request
Response with cookie
Name = Joyee
Subsequent request with cookie
Name = Joyee
Application

Cookies

- Có 2 loại Cookies
  - Thứ nhất là cookies session được lưu trong bộ nhớ trình duyệt và gửi đi thông qua header mỗi khi request.
  - Thư hai là cookie lưu trữ trong tệp tin văn bản năm trên máy người dùng. Loại này phù hợp khi bạn cần lưu trữ thông tin lâu dài
- Ví dụ: tạo cookies lưu trữ tên người dùng và lần truy cập cuối cùng trong 2 ngày

```
Response.Cookies ["UserName"].Value = "John";
Response.Cookies ["UserName"].Expires = DateTime.Now.AddDays(2);
Response.Cookies ["LastVisited"].Value = DateTime.Now.ToString();
Response.Cookies ["LastVisited"].Expires = DateTime.Now.AddDays(2);
```

Cookies

- Đọc cookies

```
if (Request.Cookies["UserName"].Value != null)
{
string Name = Request.Cookies["UserName"].Value;
}
```

- Trong trường hợp lưu trữ nhiều giá trị thì có thể đọc như sau

```
if (Request.Cookies["UserInfo"] != null)
{
string Name = Request.Cookies["UserInfo"]["UserName"];
stringVisitedOn = Request.Cookies["UserInfo"]["LastVisited"];
}
```

- UserInfo là tên cookie chứa hai sub-key có tên UserName và LastVisited.

TempData

- TempData: là một dictionary lưu trữ dữ liệu tạm theo cặp key-value. Dữ liệu có thể được truyền giữa request hiện tại và các subrequest.
- Ví dụ

```
public class HomeController : Controller
{
        public ActionResult Index()
        {
                TempData["tempText"] = "Welcome to MVC";
                return RedirectToAction("Browse");
        }
        public ActionResult Browse()
        {
                return View();
        }
}
```

Application State

- Application state cho phép lưu trữ thông tin ứng dụng theo cặp **key-value**.
- Khi người dùng truy cập bất kỳ URL nào, Application State được tạo lần đầu tiên, sau đó nó lưu trữ thông tin ứng dụng và chia sẽ cho tất các trang và user session của ứng dụng.
  - Application State sẽ hủy khi ứng dụng ngừng hoạt động
  - Application State xuất phát từ lớp HttpApplicationState.
- Khi ứng dụng chạy các sự kiện sẽ xuất hiện, Application state sẽ được khởi tạo, các sự kiện gồm
  - Application.Start
  - Application.End
  - Application.Error

Application State

- Code cho các sự kiện sẽ được đặt trong tệp Global.asax
- Khởi tạo biến Application trong code logic
  - HttpContext.Current.Application[“tenbien"]=<value>
  - Application[“tenbien”]=<value>;
- Nhận giá trị biến Application trong code Razor
  - @App.tenbien
  - @AppState["tenbien"]
- Xóa biến Application
  - Application.Remove(“tenbien”)
- Xóa tất cả các biến Applciation
  - Application.RemoveAll()

Session State

- Session state lưu thông tin cụ thể của một phiên làm việc
- Khi một người dùng truy cập thì một **session** sẽ được tạo ra với một id riêng cho người đó
- Session State có thể được truy cập ở bất kỳ trang nào trong website, thông tin của chúng cũng được lưu theo cặp key-value giống như Application State.
- Thông tin **session** sẽ được lưu trữ trong suốt quá trình request giữa người dùng và server
- Session sẽ bị hủy khi có lệnh hủy session gọi hoặc khi không có request nào trong khoảng thời gian timeout (mặc định 20 phút).

Session State

- Khởi tạo biến Session trong code logic
  - HttpContext.Current.Session[“tenbien"]=value
  - Session[“tenbien”]=value;
- Nhận giá trị biến Session trong code Razor
  - @Session.tenbien
  - @Session["tenbien"]
- Xóa biến Session
  - Session.Remove(“tenbien”)
- Xóa tất cả các biến Session
  - Session.RemoveAll()
- Hủy session
  - Session.Abandon()

Session State

- Cấu hình các thông tin của Session trong file Web.config

```
<sessionState
    cookieless="UseCookies" cookieName="Test_SessionID"
    timeout="20"
    mode="InProc" />
```

Tối ưu hiệu suất ứng dụng web

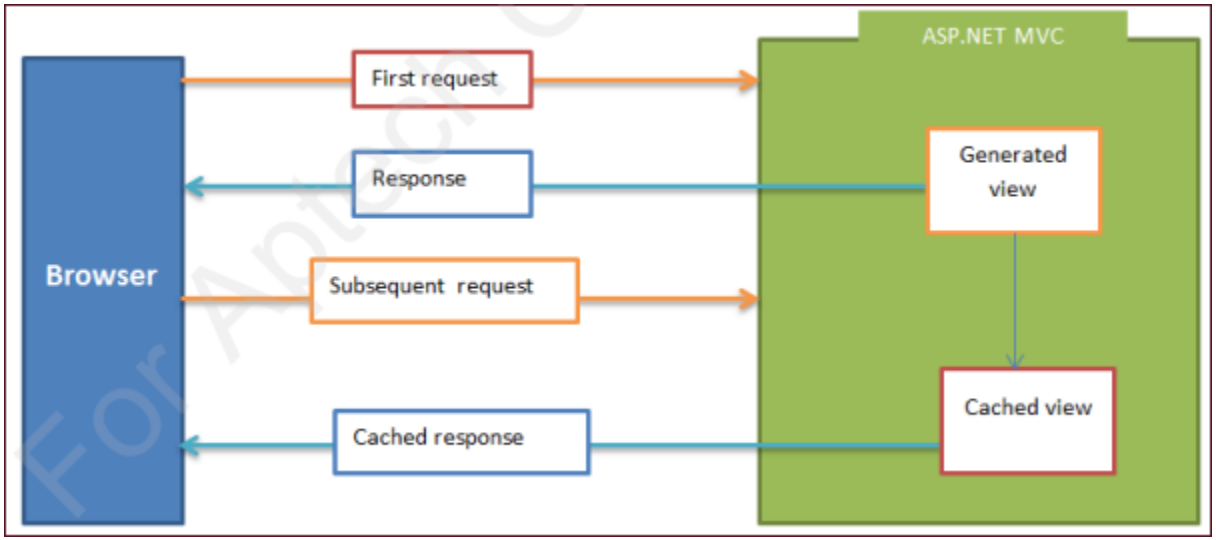
- Web application được sử dụng cho các mục đích khác nhau như
  discussion forums, online shopping, and social networking.
- Để cải thiện hiệu suất của ứng dụng bạn cần giảm sự tương tác giữa server và database.
- ASP.NET MVC Framework cung cấp các công nghệ khác nhau như:
  - Caching giúp giảm sự tương tác giữa server và database
  - Bundling and Minification giúp giảm số request và kích thước request giữa client và server.

Output Caching

- Output caching cho phép bạn cache nội dung trả về của một action
- Nếu nhiều request cùng một kết quả sinh ra của action thì asp.net sẽ lấy trong cache thay vì lấy dữ liệu từ nguồn và sinh ra
- Trước khi thực thi cache nên lưu ý:
  - Nếu nội dung thay đổi thường xuyên thì cache không hữu dụng.
  - Kiểm tra khoảng thời gian output cache bao nhiêu là phù hợp, điều này phụ thuộc vào sự thay đổi của dữ liệu có thường xuyên không?
  - Kiểm tra vị trí lưu cache ở đâu thì phù hợp.

Output Caching

- Cách thức hoạt động



- Code

```
public class HomeController : Controller
{
    [OutputCache(Duration =5)]
    0 references | 0 requests | 0 exceptions
    public ActionResult Index()
    {
        ViewBag.Message = "This page is cashed for " + DateTime.Now;
        return View();
    }
```

Output Caching

- Chỉ ra vị trí cache: các vị trí cache là Server, Proxy Server, Web Browser
- Bạn có thể chỉ ra vị trí cụ thể như sau:
  - Any: output cache ở bất kỳ đâu.
  - Client: output cache ở trình duyệt.
  - Downstream: output cache ở bất kỳ thiết bị nào hỗ trợ khả năng "HTTP cache-capable" ngoại trừ server.
  - Server: output cache trên server
  - None: không output cache

Data Caching

- Gần đây, hầu hết các ứng dụng web chứa dữ liệu động, các dữ liệu đó nhận từ database. Do vậy khi các người dùng khác nhau truy vấn database dẫn đến giảm hiệu suất ứng dụng. Để vượt qua vấn đề này bạn nên sử dụng công nghệ data caching.
- Trong ASP.NET MVC hỗ trợ lớp MemoryCache để thực hiện việc data caching

```
Customer dtUser = System.Runtime.Caching.MemoryCache.Default.
AddOrGetExisting("UserData", getUser(),
System.DateTime.Now.AddHours(1));
```

- Bạn có thể sử dụng công nghệ HTTP cache trong trình duyệt và proxy cache. Dữ liệu lưu trữ trong trình duyệt sẽ giảm việc download nội dụng từ server lặp đi lặp lại

Bundling and Minification

- ASP.NET MVC Framework cung cấp 2 công nghệ là Bundling và Minification.
- Chúng giúp cải thiện thời gian tải trang web về client, tức là giảm lượng request giữa trình duyệt và server cũng như giảm kích thước tài nguyên tải về.

Bundling

- Bundling là cách bạn bó gọn tất cả những tệp tin cần tải về và tải trong 1 lần thay vì tải nhiều tệp rời rạc
- ASP.NET MVC cung cấp lớp BundleConfig trong thư mục **App_Start** đã code sẵn việc bó gọn một số các tệp javascript và css rồi, bạn có thể mở ra bổ sung chỉnh sửa.

```
public static void RegisterBundles(BundleCollection bundles) {
bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.unobtrusive*",
                        "~/Scripts/jquery.validate*"));
bundles.Add(new tyleBundle("~/Content/css").Include("~/Content/Site.css",
"~/Content/styles.css"));
}
```

Using Bundling

- Để chèn các tệp đó vào view bạn cần dùng phương thức @Styles.Render and @Scripts.Render

```
@Scripts.Render("~/bundles/jqueryval")
@Styles.Render("~/Content/css")
```

Using Minification

- Minification cho phép bạn giảm kích thước và xóa những khoảng trắng, comment, ...trong các tệp tin css, javascript giúp thời gian tải về client nhanh hơn. Các tệp tin nay thường có phần cuối là “-min”.
- Ví dụ

  Jquery-min.js, bootstrap-min.js, bootstrap-min.css,....

Hỏi Đáp