Phiên làm việc và tối ưu hiệu năng

Nội dung

- Khái niệm về kiểm soát trạng thái (State Management)
- Khái niệm Cookies và sử dụng Cookies để kiểm soát trạng thái truy nhập
- Khái niệm về Caches và ứng dụng để tăng hiệu năng ứng dụng web
- Đóng gói và thu gọn ứng dụng

Kiểm soát trạng thái

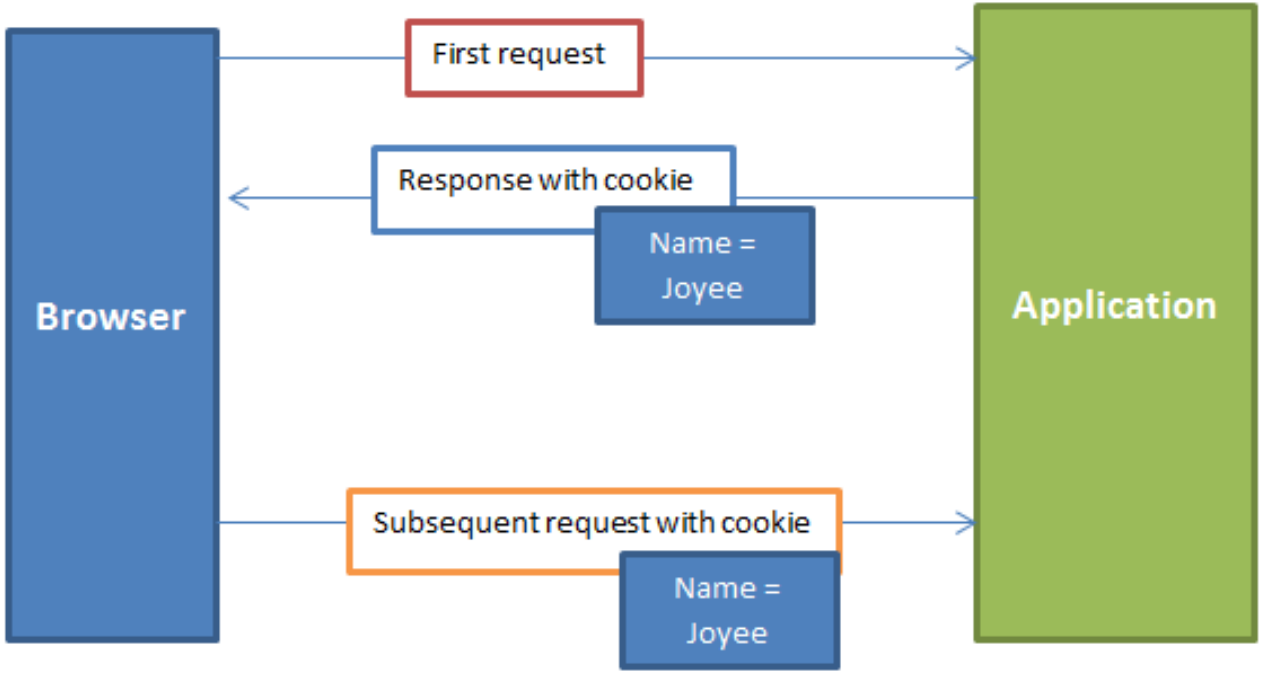
- Mô hình web sử dụng giao thức HTTP dể truyền thông giữa web client và web server
- Giao thức HTTP là giao thức phi trạng thái cho nên không có cơ chế tự động xác định các request xuất phát từ một máy client hay từ các máy client khác nhau
- Nếu có nhu cầu quản lý và kiểm soát truy nhập ta cần sử dụng các thành phần sau:
  - Cookies
  - TempData
  - Application State
  - Session State

Cookies

- Cookies được sử dụng để lưu trữ những mẩu thông tin nhỏ liên quan đến máy tính người dùng, như địa chỉ IP, loại trình duyệt, hệ điều hành và các trang web được truy cập lần cuối.
- Mục đích của việc lưu trữ thông tin này để gợi ý cá nhân hóa cho người dùng.
- Sau khi client request, Cookies được gửi về máy tính khách cùng với response
- Những cookie này được lưu trữ trên máy khách client.
- Lần sau, khi máy khách gửi yêu cầu request tới cùng trang trước, nó sẽ gửi cookie cùng với thông tin yêu cầu.
- Web server đọc cookie và trích xuất giá trị của nó. Sau đó, xử lý trang Web theo thông tin có trong cookie và gửi về response về cho máy khách.

Cookies

- Có hai loại cookie:
  - Cookie phiên được lưu trữ tạm thời trong bộ nhớ của trình duyệt được truyền lên máy chủ trong mỗi lần gửi yêu cầu request.
  - Cookie vĩnh cửu được lưu trữ trong các tệp văn bản trên máy tính của người dùng. Loại cookie này hữu ích khi bạn cần lưu trữ thông tin trong thời gian dài



Cookies

- Đoạn mã sau mô tả cách tạo Cookies

```
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);
```

- Đoạn mã tạo ra 2 Cookies là: UserName và LastVisited. Trong đó, UserName lưu tên của User, còn LastVisited lưu thời gian của lần truy cập gần nhất. Cả hai Cookies trên được thiết lập thời gian hết hạn là sau 2 ngày
- Có thể truy nhập và đọc Cookies bằng đối tượng Request
- Có thể sửa đổi Cookies bằng đối tượng Response

Cookies

- Đoạn mã sau mô tả cách truy nhập Cookies đơn

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

- Cũng có thể truy nhập Cookies phức hợp có nhiều giá trị như sau:

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

TempData

- TempData:
  - Là một từ điển lưu trữ dữ liệu tạm thời dưới dạng các cặp khóa-giá trị.
  - Cho phép lưu trữ giá trị giữa các lần request khác nhau.
- Có thể thêm các giá trị vào TempData bằng cách thêm chúng vào TempData collection.
- Dữ liệu được trong TempData tồn tại trong lần request hiện tại và tiếp theo.
- TempData rất hữu ích trong các tình huống cần truyền dữ liệu cho view trong khi chuyển hướng trang.

TempData

- Sau đây là đoạn mã của Controller Home

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

- Trong đoạn mã trên, khi gọi action Index(), chuỗi “Welcome to MVC” được lưu lại trong TempData bằng key “tempText”. Sau đó chương trình tự động điều hướng sang action Browse()

TempData

- Đoạn mã sau viết trong View của action Browse cho phép hiển thị dữ liệu trong TempData với key là “tempText”

```
@{var msg = TempData["tempText"] as String;}
@msg
```

- Ví dụ trên cho thấy dữ liệu trong TempData vẫn tồn tại mặc dù request được điều hướng sang action khác

Application State

- Application State cho phép lưu trữ thông tin dành riêng cho ứng dụng dưới dạng cặp giá trị khóa.
- Khi người dùng truy cập bất kỳ URL nào, trạng thái ứng dụng được tạo lần đầu tiên. Sau đó, nó lưu trữ thông tin dành riêng cho ứng dụng.
- Thông tin này có thể được chia sẻ với tất cả các trang và phiên người dùng của ứng dụng.
- Để làm được điều này, ta sử dụng lớp HttpApplicationState.
- Dữ liệu này được truy cập bằng cách sử dụng thuộc tính Application của đối tượng HttpContext

Application State

- Khi có bất kỳ sự kiện ứng dụng nào xảy ra, trạng thái ứng dụng sẽ được khởi tạo và thao tác.
- Những sự kiện ứng dụng này bao gồm:
  - Application.Start: Sự kiện xảy ra khi một ứng dụng nhận được yêu cầu cho một trang lần đầu tiên và khi server hoặc ứng dụng được khởi động lại. Trình xử lý sự kiện của sự kiện này chứa mã để khởi tạo các biến Application State.
  - Application.End: Sự kiện xảy ra khi máy chủ hoặc ứng dụng bị dừng hoặc khởi động lại. Trình xử lý sự kiện của sự kiện này chứa mã để xóa các tài nguyên mà ứng dụng đã sử dụng.
  - Application.Error: Sự kiện xảy ra khi xảy ra lỗi chưa xử lý.
- Trình xử lý cho ba loại sự kiện này được xác định trong tệp Global.asax.
- Có thể viết mã xử lý Application State tại một trong các trình xử lý sự kiện này.

Application State

- Để lưu trữ thông tin dành riêng cho ứng dụng ở Application State, cần tạo các biến và đối tượng. Sau đó, thêm các biến và đối tượng này vào Application State
- Những biến và đối tượng này có thể truy cập được bởi bất cứ thành phần nào của ứng dụng ASP.NET MVC.
- Bạn cần viết mã để khởi tạo các biến và đối tượng ứng dụng này bên trong hàm Application_Start() có sẵn trong tệp Global.asax.
- Đoạn mã sau cho thấy việc tạo một biến có tên TestVariable và lưu nó ở Application State:

```
HttpContext.Application ["TestVariable"] = "Welcome to MVC";
```

Application State

- Khi bạn chạy ứng dụng có biến mới được tạo và lưu trong Application State, bất kỳ trang nào của ứng dụng này đều có thể truy xuất giá trị của biến này.
- Đoạn mã cho thấy cách truy cập giá trị của TestVariable:

```
stringval = (string)HttpContext.Application["TestVariable"];
```

- Đoạn mã trên truy cập giá trị trong TestVariable và lưu trữ nó trong một biến cục bộ có tên val.
- Khi bạn đã tạo và thêm một biến hoặc đối tượng vào Application State, bạn cũng có thể xóa bỏ nó bằng phương thức Remove ().

Application State

- Đoạn mã sau đây cho thấy cách xóa biến ứng dụng có tên TestVariable khỏi Application State

```
HttpContext.Application.Remove("TestVariable");
```

- Cũng có thể sử dụng phương thức RemoveALL() để xóa tất cả các biến có sẵn trong Application State.
- Đoạn mã theo dõi hiển thị bằng cách sử dụng phương thức RemoveALL():

```
HttpContext.Application.RemoveAll();
```

Session

- Session lưu trữ thông tin mô tả cụ thể về phiên làm việc cho ứng dụng ASP.NET MVC.
- Tuy nhiên, phạm vi của các thông tin về phiên làm việc được giới hạn tồn tại trong phiên hiện thời của trình duyệt hiện tại.
- Khi nhiều người dùng truy cập đến một ứng dụng đồng thời, thì mỗi người dùng này sẽ có trạng thái phiên khác nhau.
- Giống như trạng thái ứng dụng, trạng thái phiên lưu trữ dữ liệu dành riêng cho phiên hiện thời theo từng cặp khóa – giá trị (key – value)
- Những thông tin cụ thể về phiên này nên được duy trì giữa các chuyến đi khứ hồi giữa máy chủ và máy khách qua các lần request cho các trang.

Session

- Một phiên hoạt động được xác định và theo dõi bởi một chuỗi Session Id duy nhất là một chuỗi mã ASCII.
- ASP.NET MVC Framework cung cấp lớp SessionStateModule cho phép bạn tạo và lấy chuỗi Session Id
- Giống như trạng thái ứng dụng, bạn cũng có thể lưu trữ các đối tượng và biến trong trạng thái phiên.
- Một biến phiên vẫn tồn tại trong 20 phút sau tương tác cuối cùng của người dùng.
- Bạn có thể sử dụng các phương thức tương tự như trạng thái ứng dụng, để thêm và truy cập biến hoặc các đối tượng từ trạng thái phiên.

Session

- Đoạn mã sau đây mô tả thêm biến TestVariable vào trạng thái phiên

```
Session["TestVariable"]="Welcome to MVC Application";
```

- Đoạn mã sau mô tả cách đọc biến TestVariable vừa mới tạo trong trạng thái phiên

```
stringval = (string) Session["TestVariable"];
```

Session

- Khi sử dụng trạng thái phiên, bạn cần xem xét các vấn đề sau:
  - Biến hoặc đối tượng được thêm vào trạng thái phiên vẫn tồn tại cho đến khi người dùng đóng cửa sổ trình duyệt. Ngầm định, biến hoặc đối tượng sẽ tự động bị xóa khỏi trạng thái phiên sau 20 phút nếu người dùng không request lại.
  - Biến hoặc đối tượng được thêm vào trạng thái phiên có liên quan đến một người dùng cụ thể.
- Trạng thái phiên có thể được truy cập toàn cục (global) bởi người dùng hiện tại.
- Trạng thái phiên có thể bị mất trong các trường hợp sau:
  - Người dùng đóng hoặc khởi động lại trình duyệt.
  - Người dùng truy cập vào trang Web thông qua một cửa sổ trình duyệt khác.
  - Không có request lại cho đến khi hết hạn thời gian
  - Phương thức Session.Abandon () được gọi trong mã trang.

Session

- Để xóa một đối tượng hoặc một biến đã được thêm vào trạng thái phiên, bạn có thể sử dụng các phương thức Remove() hoặc RemoveALL ().
- Khi sử dụng trạng thái phiên trong một ứng dụng, bạn cần cấu hình nó trong tệp Web.config của ứng dụng.
- File Tệp Web.config cho phép bạn xác định các tùy chọn nâng cao, chẳng hạn như thời gian chờ và chế độ trạng thái phiên.
- Đoạn mã mô tả các tùy chọn có thể cấu hình bên trong phần tử <sessionState>:

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

- Ta có thể sử dụng thuộc tính cookiless để chỉ định xem cookie có được sử dụng hay không.
- Bảng theo dõi liệt kê các giá trị mà bạn có thể chỉ định cho thuộc tính cookiless:

Giá trị	Mô tả
UseCookies	Chỉ định rằng cookie luôn được sử dụng
UseUri	Chỉ định rằng cookie không bao giờ được sử dụng.
UseDeviceProfile	Chỉ định rằng ứng dụng sẽ kiểm tra Khả năng trình duyệt để xác định xem có nên sử dụng cookie hay không.
AutoDetect	Chỉ định rằng cookie nên được sử dụng nếu trình duyệt hỗ trợ chúng.

Session

- Thuộc tính thời gian chờ của trạng thái phiên để chỉ định khoảng thời gian tính bằng phút.
- Thuộc tính mode để chỉ định nơi các giá trị của trạng thái phiên sẽ được lưu trữ.
- Có thể sử dụng một trong các giá trị sau của thuộc tính mode:
  - Custom: Chỉ định rằng một kho lưu trữ dữ liệu tùy chỉnh nên được sử dụng để lưu trữ dữ liệu trạng thái phiên.
  - InProc: Chỉ định rằng dữ liệu sẽ được lưu trữ trong cùng tiến trình với ứng dụng. Sự kiện Session.End chỉ được nêu ra trong chế độ này.
  - Off: Chỉ định rằng trạng thái phiên bị tắt.
  - QueryServer: Chỉ định rằng trạng thái phiên sẽ được lưu trữ bằng cách sử dụng cơ sở dữ liệu SQL Server để lưu trữ dữ liệu trạng thái.
  - StateServer: Chỉ định rằng trạng thái phiên sẽ được lưu trữ trong một dịch vụ chạy trên máy chủ hoặc máy chủ chuyên dụng.