

# **Licenciatura en Gestión Tecnológica**

## **Programación Avanzada II**



## **Pasaje Datos**

**Ing. Mariano Juiz**

# Agenda

1. Introducción
2. QueryString (GET)
3. Request.Form (POST)
4. Variables de Sesión
5. Variables de Aplicación
6. Viewstate
7. Cookies

# Introducción : HTTP Request y Response

## Páginas web

### Cliente

http://www.unlam.edu.ar

Internet DNS



IP= 200.47.130.101 Puerto: 80



HTTP Request

### Servidor



www.unlam.edu.ar

IP = 200.47.130.101

inicio.html

```
<html>
```

```
<body>
```

```
.....
```

```
.....
```

```
</body>
```

```
</html>
```

HTTP Response

# Introducción: Get y Post

Method = GET

```
<form method="get">
```

...

```
</form>
```

GET **/calcular.aspx?op1=2&op2=2** HTTP/1.1

...

Connection: Keep-Alive  
[blank line]

**El navegador envia los datos  
ingresados como una  
cadena de consulta**

Method = POST

```
<form method="post">
```

...

```
</form>
```

POST **/calcular.aspx** HTTP/1.1

...

Content-Type: ...  
Content-Length: 11  
[blank line]  
op1=2&op2=2

**El navegador envía los datos  
ingresados en el cuerpo  
de la solicitud HTTP**

Cualquiera sea el método utilizado, es decir GET o POST, cuando un form es enviado al servidor, decimos que se produjo un POSTBACK

# QueryString (GET)

- Se envían por el método GET
- Las variables se visualizan en la url
- Tiene un limite para escribir en algunos navegadores (255 caracteres)

## Recomendaciones

- Usarlo para enviar datos que no sean importantes
  - Opciones de Menú, filtros, etc.
- Usar `Server.HtmlEncode` para sustituir caracteres especiales en un texto, que el HTML no reconoce
  - Para “decodificarlo” se usa `Server.HtmlDecode`
  - Evita que se introduzcan tags de html, asp.net, etc.
- Usar `Server.UrlEncode`
  - Idem anterior, pero orientado a la url

# Request.Form (POST)

- Se envían por el método POST
- Las variables se envían de formulario a formulario, por el encabezado HTTP
- Es un poco más seguro que el GET
- Request.Form es una colección

## **Recomendaciones**

- Encriptar datos antes de enviarlos

# Variables de Sesión

- Son objetos que se inician cuando el usuario ingresa en la página y finalizan cuando deja el sitio o por timeout
- Son datos que solo visualiza el usuario en cuestión

**En Global.asax** (se inicializan los objetos)

```
protected void Session_Start(Object sender, EventArgs e)
{
    Session["usuarioID"] = String.Empty;
    Session["sexo"] = String.Empty;
}
```

**En cualquier parte de la aplicación web** (se setean los objetos)

```
Session["usuarioID"] = "Iquiroya";
Session["sexo"] = "M";
```

## Recomendaciones

- Encriptar datos antes de guardarlos

# Variables de Sesión

## **Modos**

- InProc
- State Server
- SqlServer
- Custom
- Off



# Variables de Sesión

## Modo InProc

- Es el modo por defecto (y el más óptimo)
- El estado de la sesión se almacena en la memoria del servidor web
- Ofrece el mejor rendimiento y buena seguridad
- No se persiste si se reinicia la aplicación web o a través varios servidores (Web Farm)

```
<sessionState mode="InProc" cookieless="false" timeout="20" />
```

- Cookieless define si almacena el "session Id" en el usuario o es ingresado en la querystring de la url

```
http://www.myapp.com/(55mfgh55vgblurtywsityvjq)/Resultado.aspx
```

## Recomendaciones

- No guardar muchos datos porque ocupan memoria de servidor
- Encriptar los datos antes de guardarlos

# Variables de Sesión

## StateServer

- El estado de la sesión se almacena en un servicio llamado ASP.NET State Service (aspnet\_state.exe)
- Se envían datos por el protocolo HTTP sobre un puerto TCP
- Persiste aunque se reinicie la aplicación o a través de varios servidores (Web Farm)
- Ofrece menor rendimiento que el modo InProc, pero mayor fiabilidad y escalabilidad

```
<sessionState mode="StateServer" cookieless="false"  
stateConnectionString="tcpip=myserver:42424" timeout="20" />
```

## Recomendaciones

- No detener el servicio (se pierden los datos de la sesión)
- Encriptar los datos antes de guardarlos

# Variables de Sesión

## SqlServer

- El estado de la sesión se almacena en una base de datos de SQL Server, brindando mayor estabilidad y escalabilidad
- Persiste aunque se reinicie la aplicación o a través de varios servidores (Web Farm)
- En las mismas condiciones de Hardware, ofrece menor rendimiento que **State Server** pero ofrece una mejor integridad de los datos y reporting.

```
<sessionState mode="SqlServer"
sqlConnectionString="data source=127.0.0.1;user
id=sa; password=" cookieless="false" timeout="20" />
```

## Recomendaciones

- Crear la base de datos "ASPState" usando el script "InstallState.sql" (ubicado en la carpeta *WinDir\Microsoft.Net\Framework\Version*)
- Encriptar los datos antes de guardarlos

# Variables de Sesión

## Custom

- Permite especificar un proveedor de almacenamiento de la sesión customizado
  - Ej: [https://technet.microsoft.com/es-ar/library/cc725582\(v=ws.10\).aspx](https://technet.microsoft.com/es-ar/library/cc725582(v=ws.10).aspx)
- Es necesario implementarlo

```
<sessionState mode="StateServer" cookieless="false"  
stateConnectionString="tcpip=myserver:42424" timeout="20" />
```

## Recomendaciones

- Encriptar los datos antes de guardarlos

## Off

- Deshabilita el estado de la sesión.
- Si la aplicación web no usa sesión, se mejora el rendimiento.

# Variables de Aplicación

- Son objetos que se inician con la Aplicación Web y persisten hasta detenerla
- Son datos que visualizan todos los usuarios de la aplicación web

**En Global.asax** (se inicializan los objetos)

```
protected void Application_Start(Object sender, EventArgs e)
{
    Application["localidad"] = "San Justo";
    Application["universidad"] = "UNLAM";
}
```

**En cualquier parte de la aplicación web**

```
Response.Write(Application["localidad"].ToString());
```

# Variables de Aplicación

## Recomendaciones

- Gestionar la concurrencia:
  - Application.Lock antes de actualizar
  - Application.Unlock después de actualizar
- ¡Cuidado con el rendimiento!
  - Los bloqueos pueden ralentizar
  - No se comparte entre distintos servidores
- Utilizar cuando son datos para todos los usuarios (provincias, localidades, etc.)

# ViewState

- Mantiene el estado de los controles de una misma página entre una ida y venida al servidor.
- Utiliza un campo oculto llamado "ViewState" con un valor incomprensible y generalmente muy largo
  - Depende de la página, de la cantidad de controles de los que haya que controlar el estado, entre otras cosas
- Es una variable del ámbito de petición una misma página
- Se inhabilita por página o por control con *EnableViewState="false"*

## Recomendaciones

- Cuando se recupere el valor del ViewState, hay que parsearlo a su tipo de dato
- Se puede trabajar con el ViewState en la misma página y siempre que se hayan realizados PostBack
- Útil para datos de pequeña longitud (porque ocupa ancho de banda)<sup>15</sup>

# Cookies

- Se guarda información en el disco rígido del cliente
- Están asociadas a un sitio web y no a una página en particular
- Son útiles para mantener la continuidad en una aplicación Web

```
Response.Cookies["usuario"]["usuarioId"] = "jquirola";  
Response.Cookies["usuario"]["ultimaVisita"] = DateTime.Now.ToString();  
Response.Cookies["usuario"].Expires = DateTime.Now.AddDays(1);
```

```
lblUsuarioId.Text = Request.Cookies["usuario"]["usuarioId"];
```

## Recomendaciones

- Encriptar los datos antes de guardarlos
- Almacenar datos no susceptibles y no invalidantes para la aplicación web
- No depender de los datos guardados como cookies porque el cliente los puede tener inhabilitados o pueden ser borrados
- Almacenar poca información (tamaño máximo de 4096 bytes, 20 cookies por sitio y/o 300 cookies en total)



# **Licenciatura en Gestión Tecnológica**

## **Programación Avanzada II**



# **Muchas Gracias**

**Ing. Mariano Juiz**