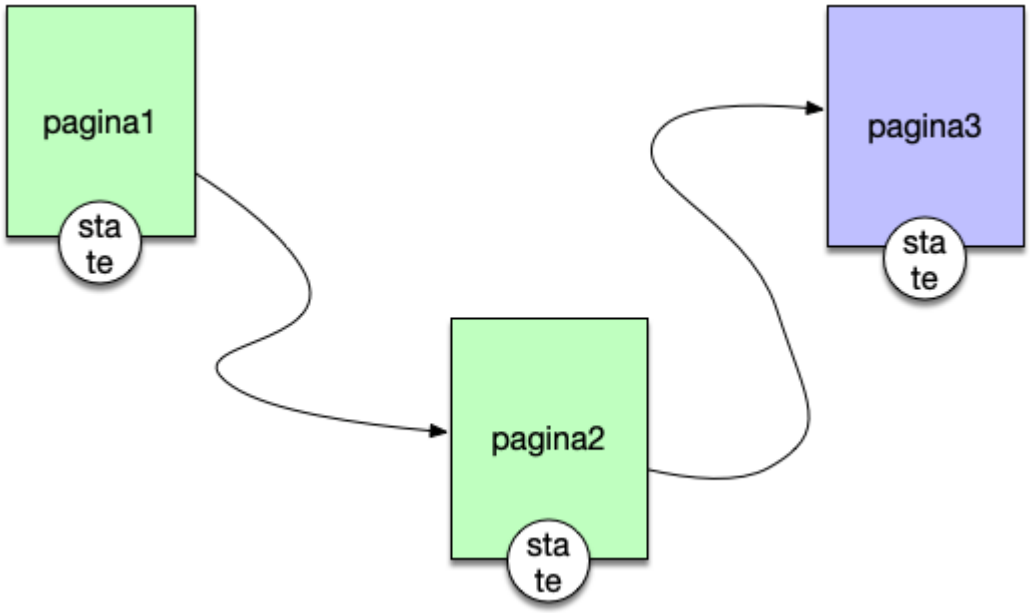
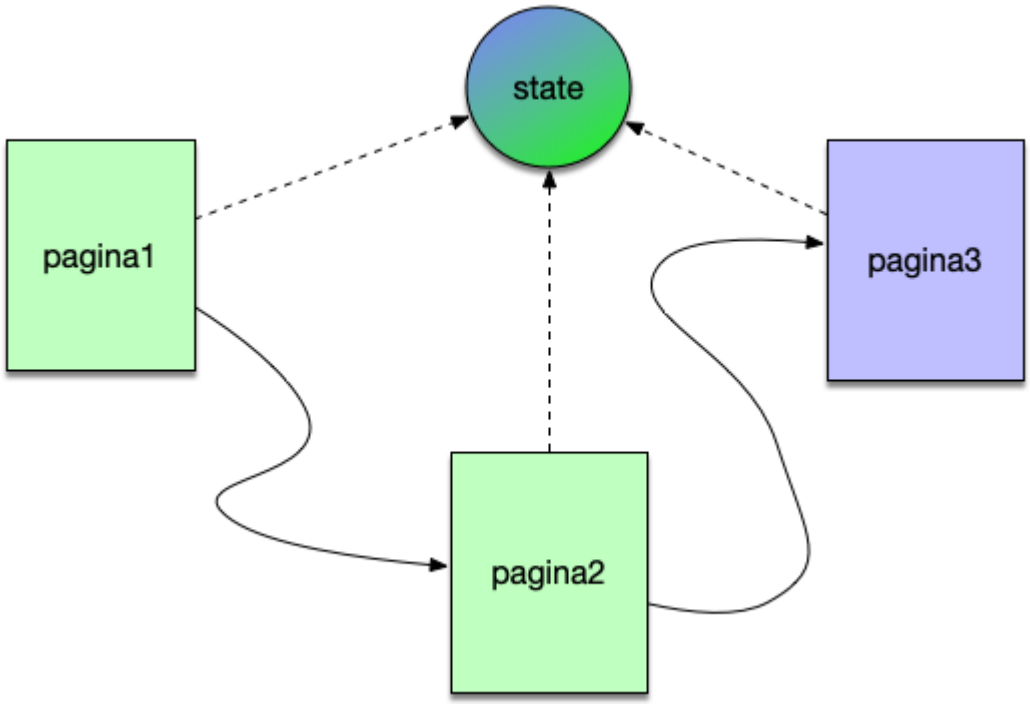
El uso del concepto de httpSession es muy común. Las sesiones Http permiten normalmente el almacenamiento de estado entre un conjunto de páginas que el usuario utiliza de forma habitual durante su navegación. Recordemos que el protocolo Http es Stateless y cada página mantiene su propio estado. Es decir si nosotros navegamos entre páginas cada página es en principio independiente.

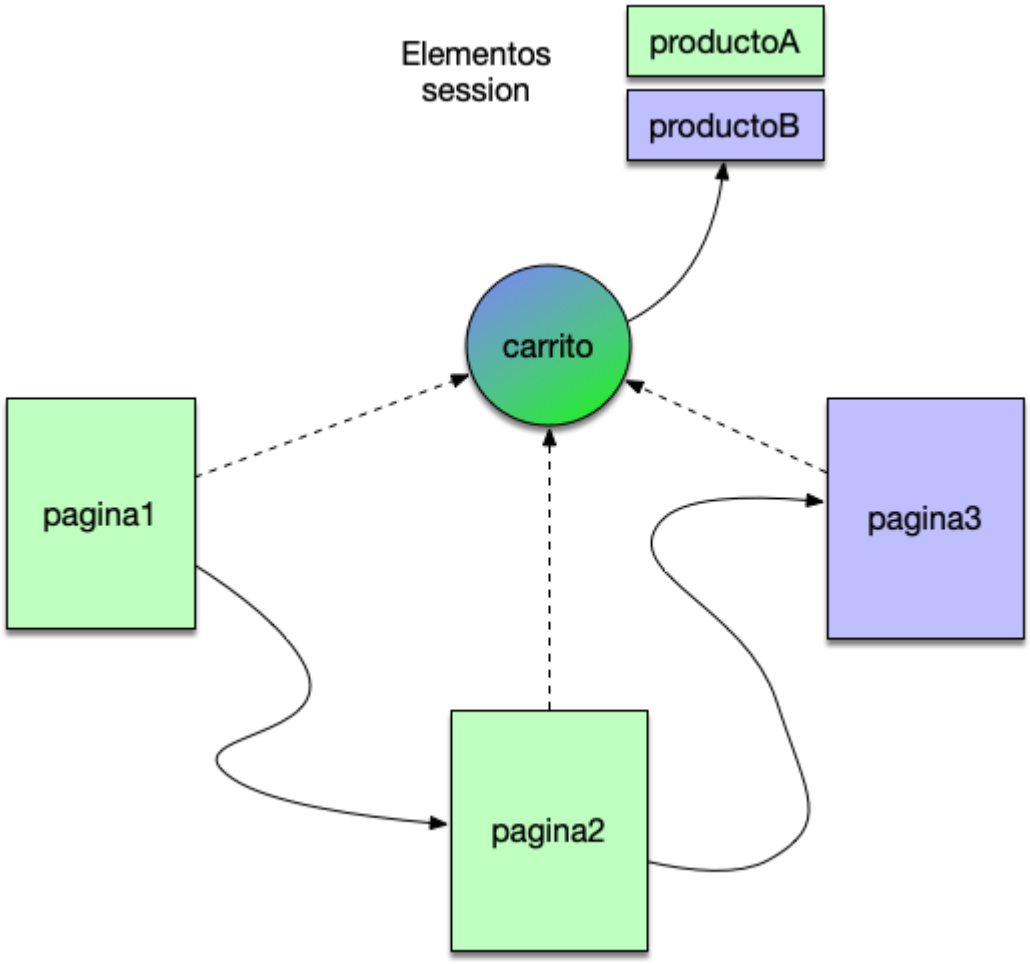


Sin embargo existen situaciones en las que nosotros necesitamos compartir estado entre varias páginas de tal forma que podamos acceder en todas ellas a la misma información. Este estado se almacena normalmente en una HttpSession.

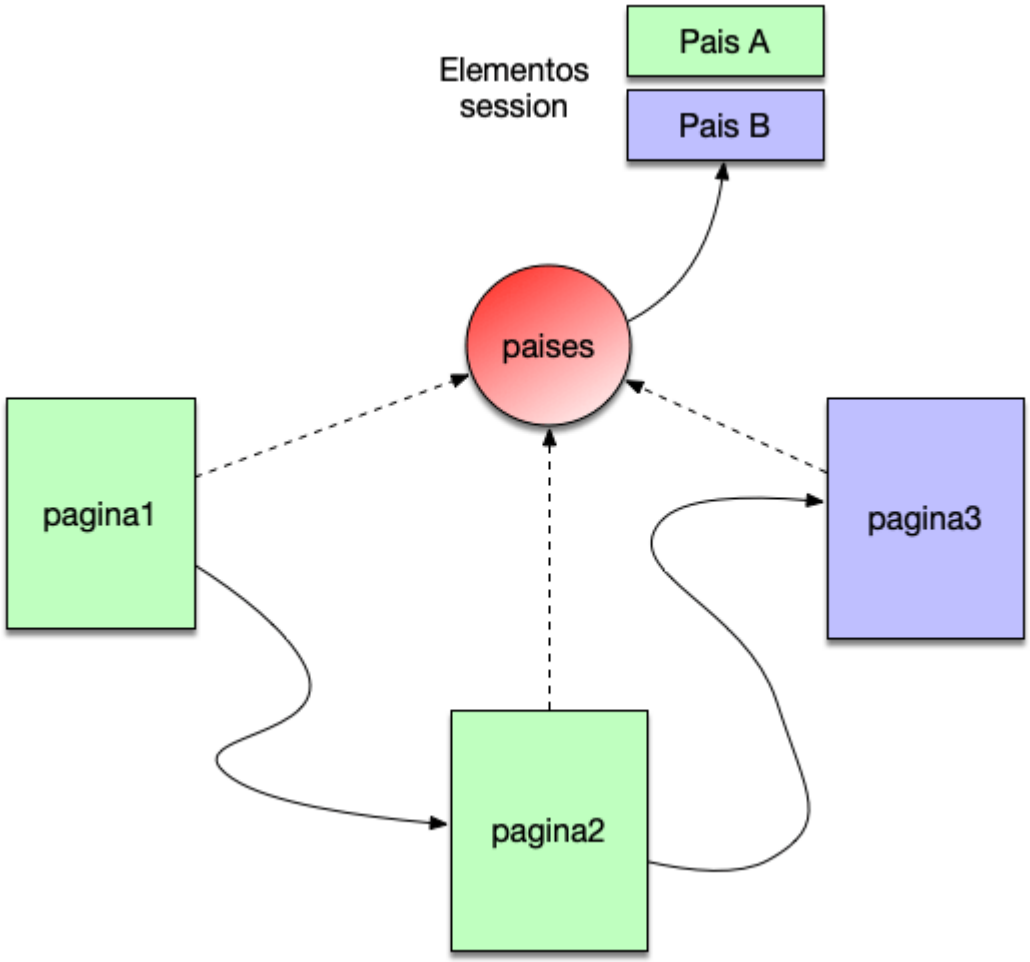


HttpSession y Carrito de la compra

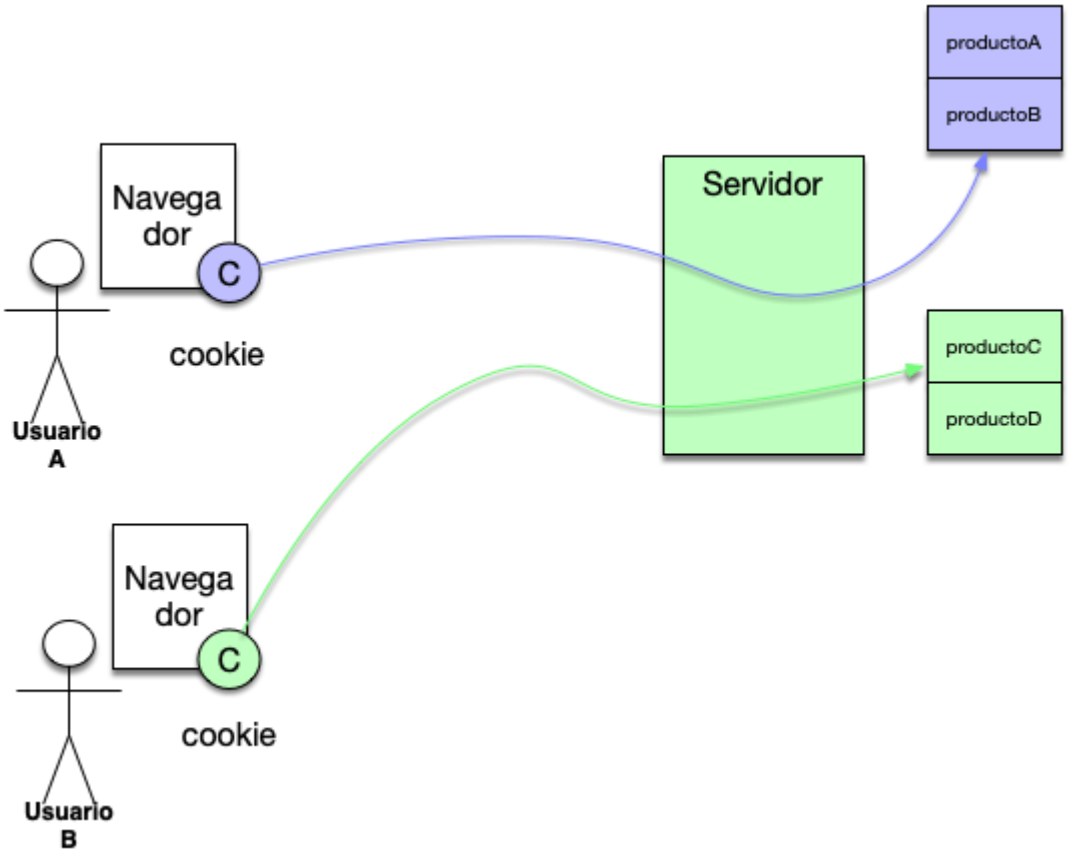
Un ejemplo clásico de esta situación es un carrito de la compra en el cual el usuario va añadiendo elementos a su sesión y los mantiene vivos mientras va pasando página por página añadiendo productos. Hasta aquí todo es correcto y usamos el concepto de session de una forma correcta.



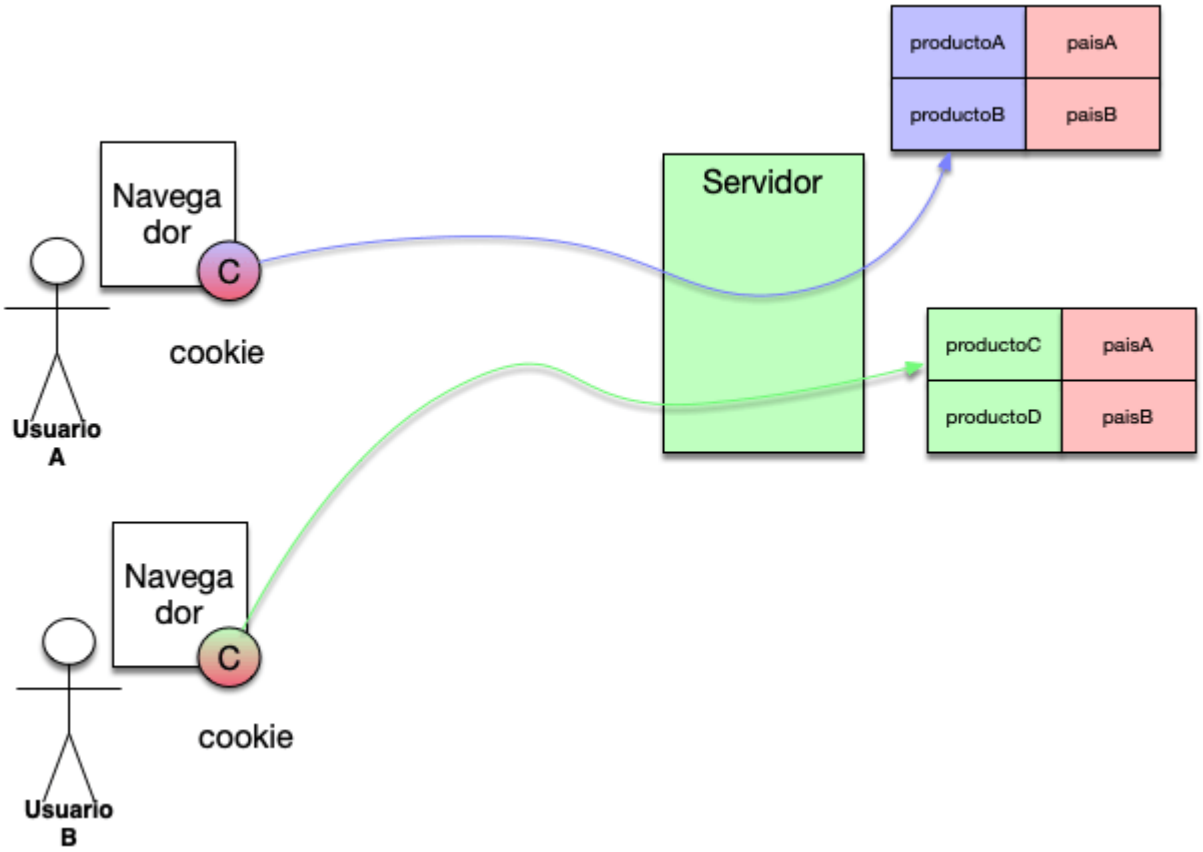
Ahora bien en muchos casos me encuentro en situaciones un poco diferentes . Las sesiones se utilizan para almacenar la información que el usuario necesita compartir entre un conjunto de páginas .¿ Es eso cierto?. La realidad es que NO . Las sesiones se usan para almacenar la información “del usuario” que se necesita compartir entre varias páginas. No cualquier tipo de información que necesitemos almacenar entre varias páginas . Demonos cuenta del matiz. El carrito de la compra es una información del usuario , pertenece a este y es única de él. En muchos casos podemos tener información que almacenamos en la sesión y que necesita ser compartida por el usuario entre varias páginas pero que no le pertenece un ejemplo clásico es un listado de paises. El listado le necesitamos presentar en varias páginas.



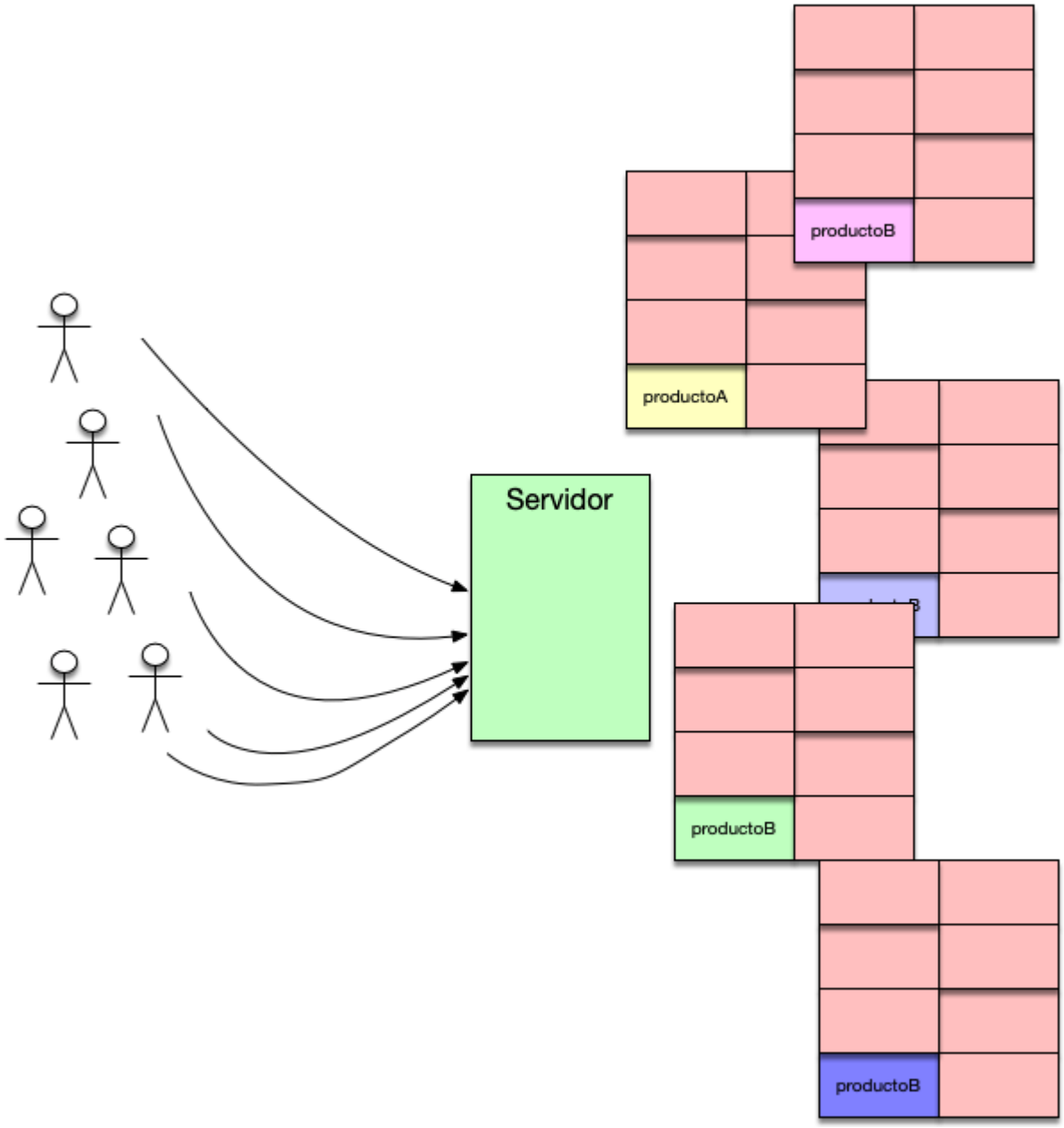
Esta es una información que almacenamos a nivel de HttpSession pero que no pertenece concretamente a este usuario. Esto es realmente un problema ya que estamos utilizando la sesión para almacenar información que no es su ubicación correcto. En este caso una lista de paises se puede almacenar en una cache u objeto singleton similar que se cargue una vez en memoria y no en la sesión. ¿Cuál es el problema , el problema es que las sesiones se almacenan en el servidor por cada usuario y tienen un tiempo de vida habitual de 30 minutos a 1 hora aunque el usuario no se conecte más. Estas sesiones quedan identificadas por una cookie que se envía al navegador y que el usuario nos remite en cada petición.



En una situacion de carrito de la compra cada usuario guarda sus productos , lo cual es correcto. Ahora bien en una situación como la de los paises cada usuario guarda la lista de paises en memoria durante la sesión por lo tanto tendremos mucha información repetida por cada usuario en la memoria del servidor y es incorrecto.

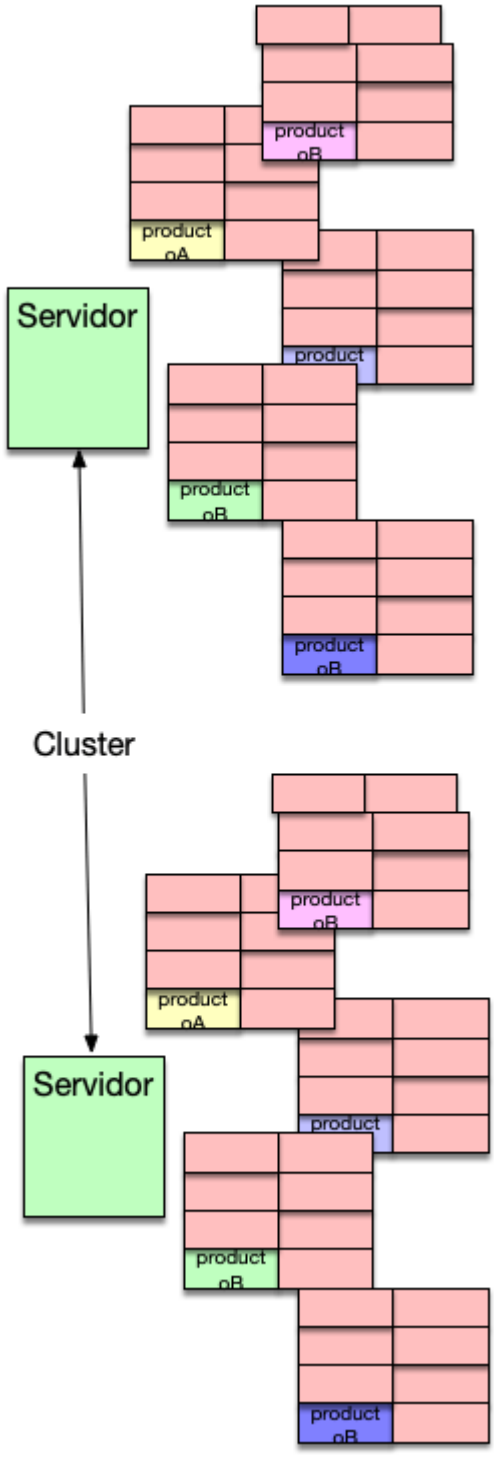


El problema se agrabara cuando poco a poco nos dejemos llevar y almacenemos más y mas información que no pertenece realmente al usuario en la sesión ya que cada usuario guardará su propia copia. Cuando nos queramos dar cuenta si tenemos muchos usuarios activos en la aplicación la situación será problemática.



Esto implicará que el servidor consumirá más y más memoria. Muchas veces la gente dice que eso no es problemático ya que dispone de memoria suficiente .Sin embargo si el número de usuarios crece o si el tiempo de vida de la sesión de cada usuario se amplia porque un cliente lo pide . Cada vez almacenaremos más y más datos repetidos en la sesión llegando un punto en el cual la memoria no será suficiente y tendremos un problema severo. Me he

encontrado en situaciones en las cuales alguien sugiere que se montemos un cluster y dupliquemos la memoria disponible. Como posible solución al problema . El problema es que en un cluster las sesiones se replican y por lo tanto estaremos en una situación similar



Si queremos que nuestras aplicaciones escalen de la forma correcta deberemos usar las sesiones de una forma “coherente” almacenando la información que “pertenece a cada

usuario” y este necesita compartir , no podemos asumir que se almacene cualquier tipo de información ya que esto limitará la escalabilidad

1. HttpSessionListener un concepto importante
2. OpenSessionInView AntiPattern y sus problemas
3. Java HttpSession Timeout
4. Usando Java Session en aplicaciones web
5. El patrón singleton