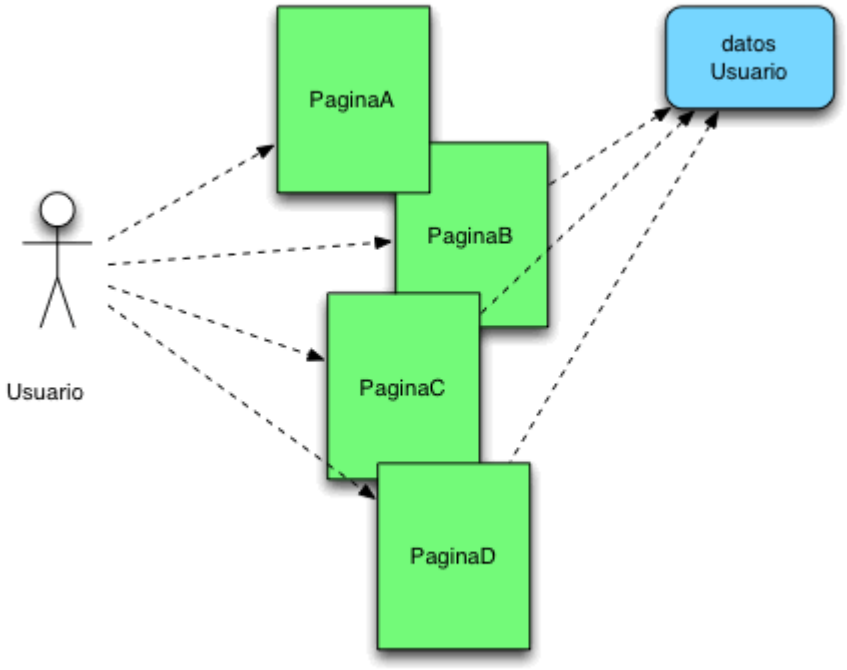
Tabla de Contenidos

- Java Session
- HttpSession y Funcionamiento
- Usuarios y Sessiones
- Un ejemplo sencillo
- Java Session otros artículos relacionados

Uno de los conceptos que más problemas produce cuando comenzamos a trabajar con aplicaciones web en Java es el concepto de java session (HttpSession) que sirve para almacenar información entre diferentes peticiones HTTP ya que este protocolo es stateless (sin estado).

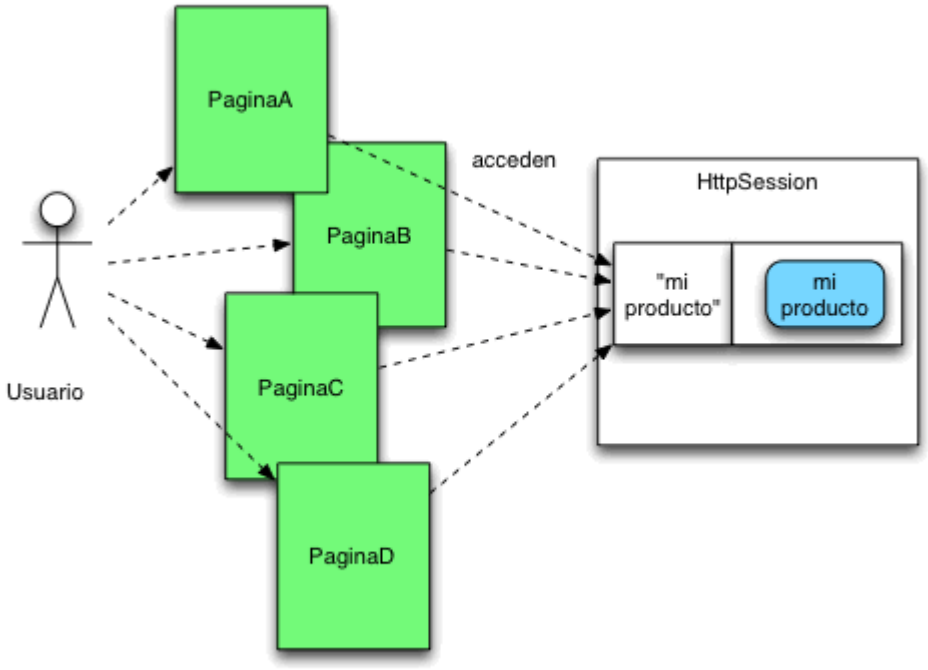


Así pues en muchas ocasiones nos encontraremos con el problema de compartir estado (datos usuario) entre un conjunto amplio de páginas de nuestra Aplicación. Un ejemplo clásico es un carrito de la compra en donde almacenamos los items que compramos.



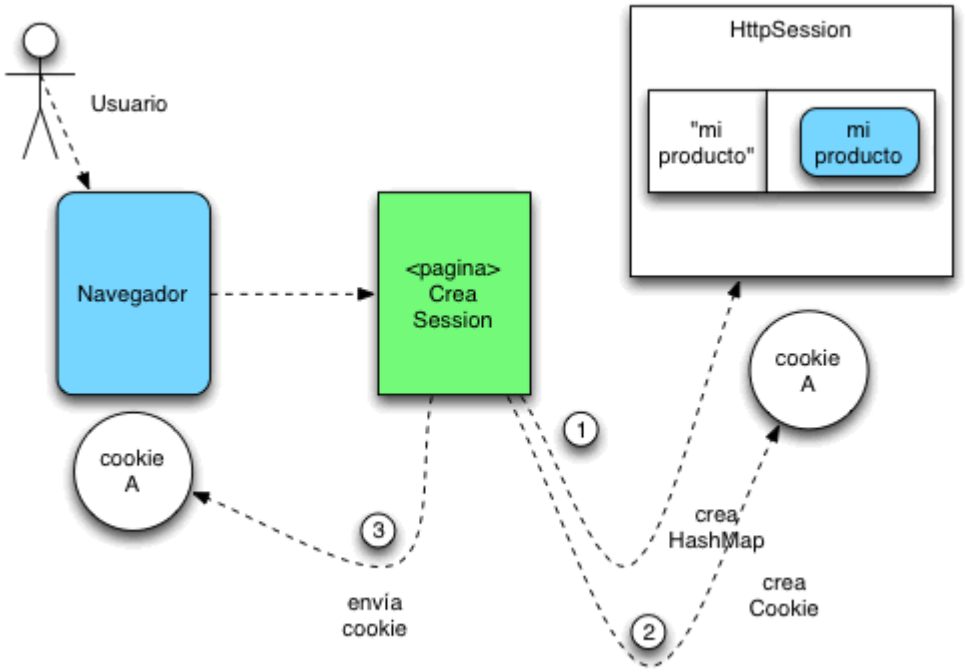
Java Session

Para solventar este problema en la plataforma Java EE se usa de forma muy habitual la clase HttpSession que tiene una estructura de HashMap (Dicccionario) y permite almacenar cualquier tipo de objeto en ella de tal forma que pueda ser compartido por las diferentes páginas que como usuarios utilizamos.



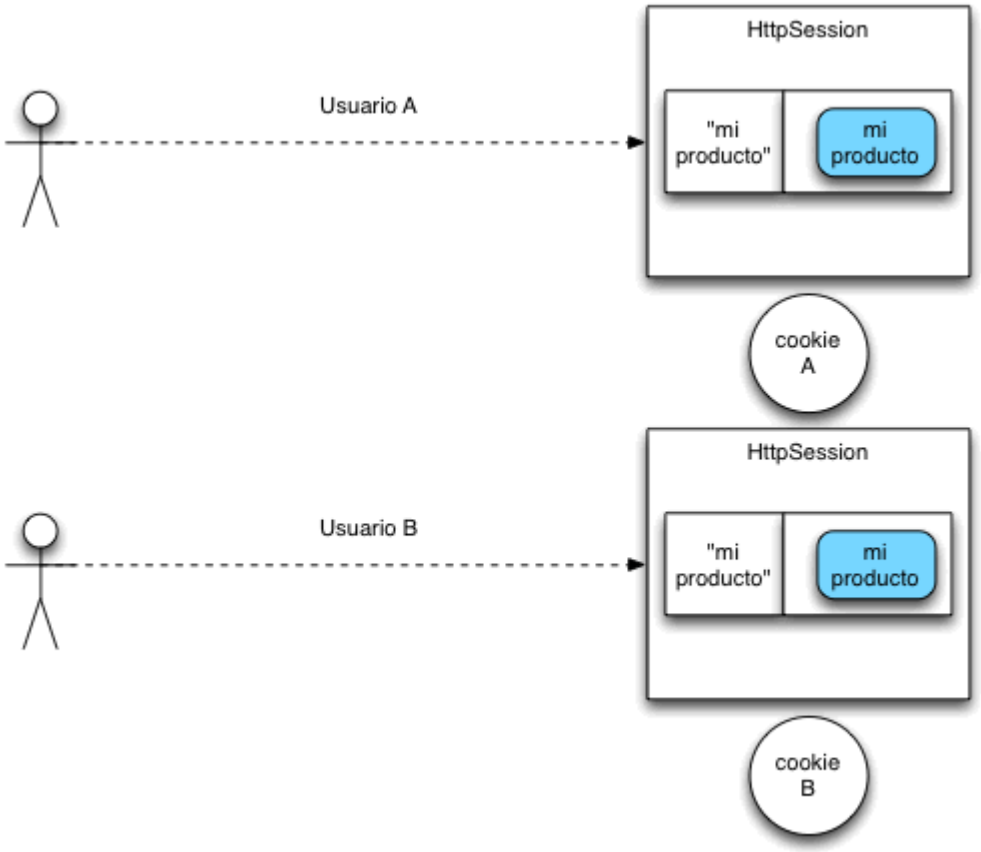
HttpSession y Funcionamiento

El funcionamiento del sistema de sessiones es relativamente sencillo. Cada vez que un usuario crea una session accediendo a una página (que la genere) se crea un objeto a nivel de Servidor con un HashMap vacío que nos permite almacenar la información que necesitamos relativa a este usuario. Realizado este primer paso se envía al navegador del usuario una Cookie que sirve para identificarle y asociarle el HashMap que se acaba de construir para que pueda almacenar información en él. Este HashMap puede ser accedido desde cualquier otra página permitiéndonos compartir información.



Usuarios y Sessiones

El concepto de Session es individual de cada usuario que se conecta a nuestra aplicación y la información no es compartida entre ellos. Así pues cada usuario dispondrá de su propio HashMap en donde almacenar la información que resulte útil entre páginas.



Un ejemplo sencillo

Vamos a ver un ejemplo utilizando dos Servlets básicos en el cual un servlet almacena datos en la sesión (concretamente un objeto producto) y otro servlet lee los datos de la session y los muestra por pantalla.

```
package com.arquitecturajava;

public class Producto {
    private int id;
    private String concepto;
    private double importe;
```

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getConcepto() {
    return concepto;
}

public void setConcepto(String concepto) {
    this.concepto = concepto;
}

public double getImporte() {
    return importe;
}

public void setImporte(double importe) {
    this.importe = importe;
}

public Producto(int id, String concepto, double importe) {
    super();
    this.id = id;
    this.concepto = concepto;
    this.importe = importe;
}
```

```
    public Producto() {
        super();
    }
}
```

Acabamos de construir la clase Java de Producto. El siguiente paso es crear un Servlet que introduzca los datos en la sesión.

```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/CrearSession")
public class CrearSession extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        HttpSession misession = request.getSession(true);
        Producto miproducto = new Producto(1, "telefono", 300);
        misession.setAttribute("producto", miproducto);
        PrintWriter pw = response.getWriter();
        pw.println("<html><body>Producto en session</body></html>");
        pw.close();
    }
```

```
}
```

Introduccido el objeto en la sessión no nos queda mas que leerlo desde otro servlet

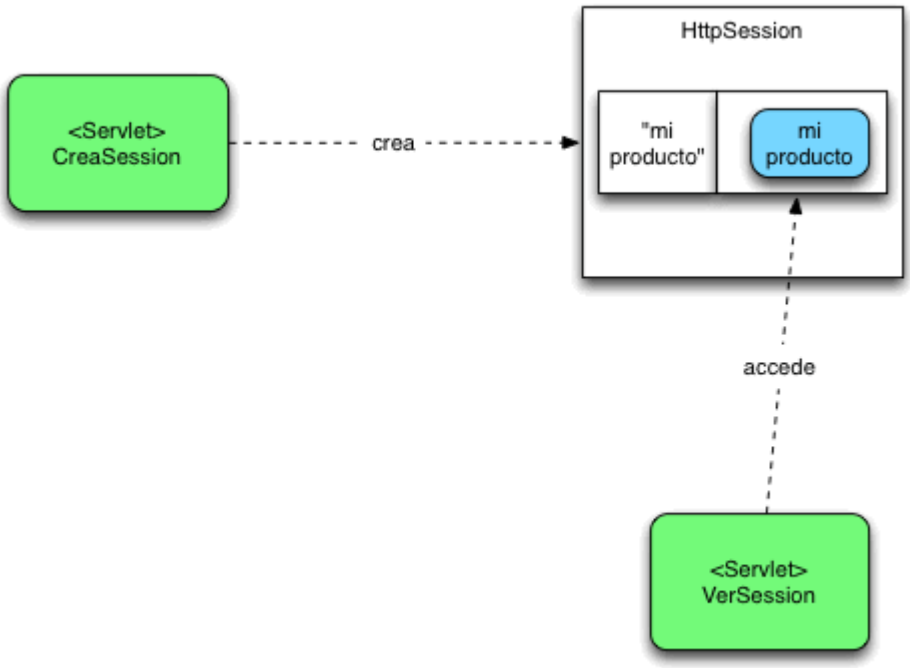
```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

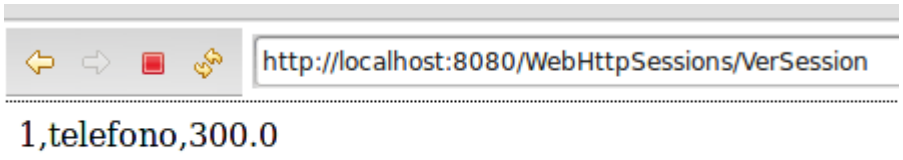
@WebServlet("/VerSession")
public class VerSession extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        HttpSession misession = request.getSession();
        Producto miproducto = (Producto)
misession.getAttribute("producto");
        PrintWriter pw = response.getWriter();
        pw.println("<html><body>" + miproducto.getId() + "," +
miproducto.getConcepto() + "," + miproducto.getImporte() +
"</body></html>");
        pw.close();
    }
}
```

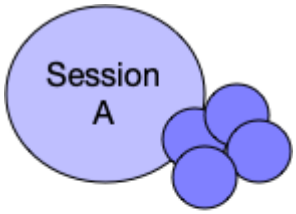
En este caso el primer Servlet (CreaSession) se encarga de crear la session y almacenar un producto en ella. Mientras el segundo Servlet se encargará de acceder a los datos y mostrarlos por pantalla.

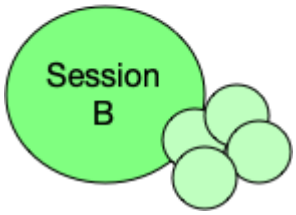


Una vez invocado el Servlet que crea la session invocamos a VerSession y nos mostrará por pantalla los datos que se han introducido. Hemos compartido información con Java Session



Hemos compartido información perteneciente al mismo usuario entre dos Servlets. Recordemos que no se debe abusar del almacenamiento de objetos en Session ya que al ser por cada usuario si tenemos muchos usuarios concurrentes estaremos obligando al servidor a utilizar mucha memoria para almacenarlos.





No solo es importante la concurrencia sino que muchas veces la session tiene un ciclo de vida x .Puede durar 30 minutos o 2 horas y los datos se mantendrán almacenados ese tiempo.

Java Session otros artículos relacionados

CURSO Introducción Patrones Diseño

GRATIS

APUNTATE!!