HttpSession Yönetimi

harezmi bilişim çözümleri

HTTP ve Oturum Yönetimi

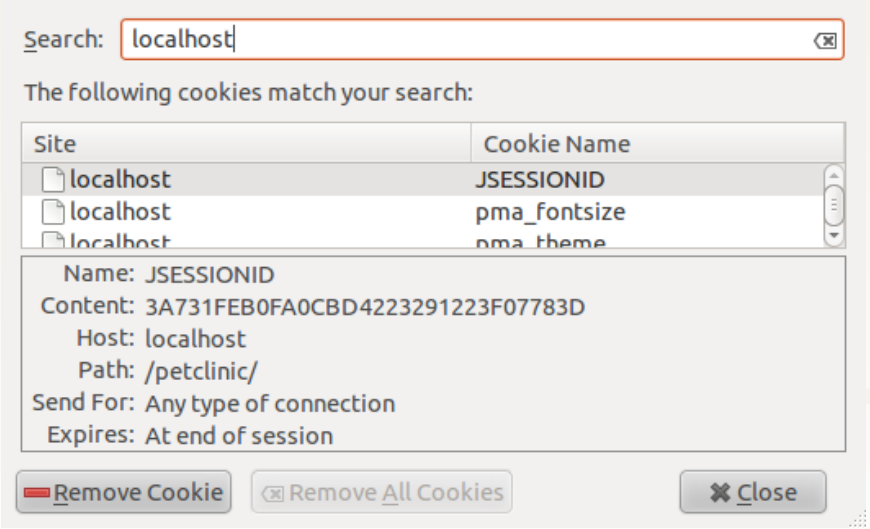
- HTTP **stateless** bir protokoldür
- Normalde aynı kullanıcının iki request'i arasında herhangi bir **state bilgisi server tarafında tutulmaz**
- HttpSession iki request arasında tutulması gereken **state bilgisinin saklanabileceği yapı**dır
- HttpSession **unique bir ID** ile ifade edilir
- Bu ID iki request arasında ya **cookie** içerisinde ya da **request parametresi** ile taşınır

Session Cookie ile Oturum Takibi

```
public class Servlet1 extends HttpServlet {
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

        HttpSession session = request.getSession();
        String message = request.getParameter("message");
        session.setAttribute("msg", message);
    }
}
```

http://localhost:8080/petclinic/servlet1?message=hello

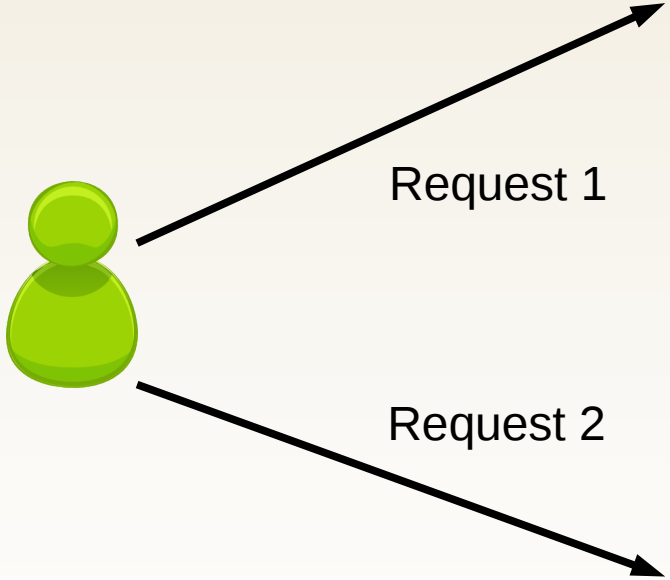
http://localhost:8080/petclinic/servlet2




```
public class Servlet2 extends HttpServlet {
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

        HttpSession session = request.getSession(false);
        String message = (String) session.getAttribute("msg");
        response.getWriter().write(message);
    }
}
```

JSESSIONID Request Parametresi ile Oturum Takibi

http://localhost:8080/petclinic/servlet1?message=hello



İkinci request'in aynı HttpSession'a erişmesini sağlamanın diğer yolu request'in **JSESSIONID request parametresi** ile çağrılmasıdır

Session ID değeri Cookie içeriğinden öğrenilebilir

httpSession.getId() metodu ile elde edilebilir

response.encodeURL() metodu ile de bir sonraki request'in URL'ine eklenebilir

http://localhost:8080/petclinic/servlet2;jsessionid=021699B3333F5ED6DD7FE78AD045216A

Session Timeout

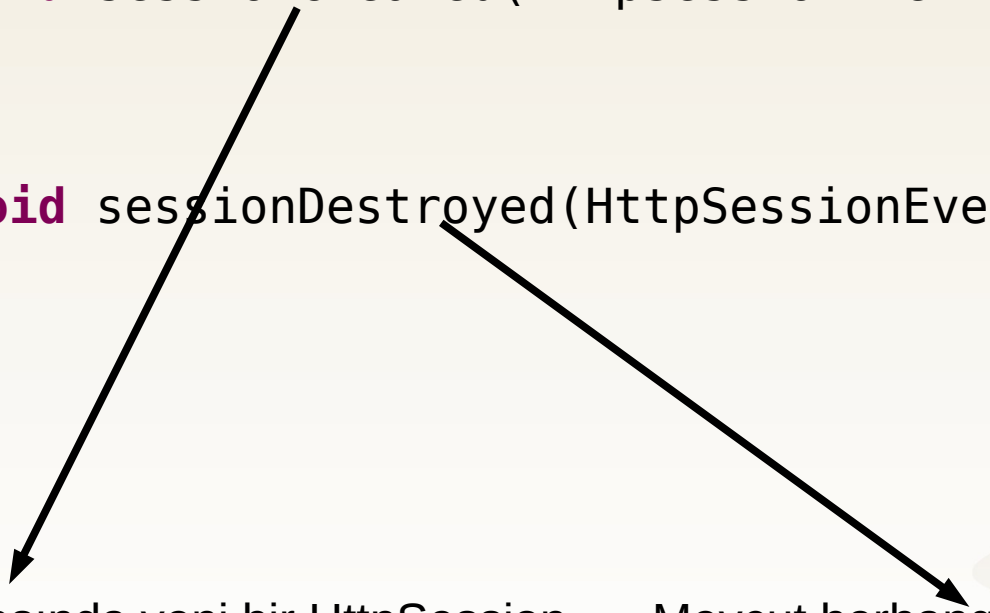
- İstemci tarafından **belirli bir süre HttpSession'a erişilmediği vakit** server tarafından Session sonlandırılır
- Bu süre sadece **web.xml içerisinden** kontrol edilebilir
- Session'ı sonlandırmanın diğer bir yolu doğrudan **HttpSession nesnesinin invalidate() metodu**nu çağırmaktır
- Varsayılan timeout değeri 30 dk'dır

HttpSessionListener Ne İşe Yarar?

- **HttpSession** nesneleri **yaratıldıkları ve sonlandırıldıkları vakit** web uygulaması içerisinde **bir takım işlemlerin yapılması** gerekebilir
- **HttpSessionListener** arayüzü bu amaçla kullanılır
- Bu arayüzü implement eden sınıflar web uygulamasındaki **HttpSession instance'ları hakkında bilgilendirilir**ler
- Devreye girebilmeleri için **web.xml içerisinde tanımlanmaları gerekir**

HttpSessionListener Örneği

```
public class MySessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent se) {

    }

    public void sessionDestroyed(HttpSessionEvent se) {

    }

}
```



Web uygulamasında yeni bir HttpSession Yaratıldığı vakit çağrılır

Mevcut herhangi bir HttpSession sonlandırıldığı Vakit çağrılır

Sonlandırma session timeout olması veya Explicit invalidate işlemi ile gerçekleşebilir

Web.xml'de Listener ve Session Timeout Konfigürasyonu

```
<web-app>

    <listener>
        <listener-class>x.y.z.MySessionListener</listener-class>
    </listener>

    <session-config>
        <session-timeout>5</session-timeout>
    </session-config>

</web-app>
```

Default değer 30 dk'dır

Web.xml'de Listener tanımları, listener elemanları içerisinde ayrı ayrı yapılmalıdır

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri