ردیابی تماس کاربر

HTTP یک پروتکل ناپایدار[1] است که در آن سرویس‌گیرنده یک ارتباط با سرویس‌دهنده برقرار کرده و اطلاعات یا منابعی را درخواست می‌کند. سرویس‌دهنده به درخواست پاسخ داده و درصورت عدم وجود منابع، پیغام خطای HTTP نمایش داده‌می‌شود. پس از این مرحله ارتباط قطع شده و سرویس‌دهنده هیچگونه اطلاعاتی راجع به سرویس‌گیرنده ندارد. بنابراین سرویس‌دهنده، درخواست بعدی همان سرویس‌گیرنده را به عنوان یک درخواست جدید و بدون ارتباط با درخواست قبلی آن فرض می‌کند. این مکانیزم باعث می‌شود که پروتکل HTTP ناپایدار نامیده شود.

یک پروتکل، پایدار[2] نامیده می‌شود اگر پاسخ به درخواست‌ها قابل برنامه‌ریزی باشد، بطوریکه نه تنها به درخواست جاری بلکه به درخواست‌های قبلی نیز وابسته باشد.

بطور معمول در یک پروتکل پایدار، درخواست‌های چندگانه سرویس‌گیرنده و پاسخ‌ها از طریق تنها یک ارتباط بین سرویس‌دهنده و سرویس‌گیرنده مبادله می‌شود. بسته به نوع ارتباط، سرویس‌دهنده می‌تواند درخواست‌های یک سرویس‌گیرنده را در قالب یک تماس[3] تعریف کند. به عنوان مثال می‌توان پروتکل FTP را نام برد. FTP یک پروتکل پایدار است، درخواست‌های چندگانه کاربر و پاسخ آن در یک session فرستاده می‌شود. در این نوع پروتکل ارتباط با دستور Open شروع و با اجرای دستور Exit با قطع ارتباط شبکه پایان می‌یابد.

سرویس‌دهنده FTP می‌تواند باتوجه به وضعیت session تصمیم بگیرد. به عنوان مثال یک سرویس‌دهنده FTP می‌تواند تعداد دریافت‌ها را در یک session محدود کند.

اما اهمیت پروتکل پایدار در چیست؟ یک پروتکل پایدار امکان پیاده‌سازی یک برنامه تحت وب را بصورت مجموعه ای از درخواست‌ها و پاسخ‌ها فراهم می‌کند. یک حساب پس‌انداز را درنظر بگیرید ، وقتی مشتری صفحه وضعیت حساب خود را درخواست می‌کند، سرویس‌دهنده باید قادر به

---

[1] Stateless
[2] State full
3 Session

تشخیص مشتری به عنوان شخص معتبر برای بانک باشد. اگر پروتکل ارتباطی ناپایدار باشد، مشتری باید اطلاعات شناسایی (اعتباری) خود را به همراه تمام درخواست‌های خود بفرستد. این موضوع برای تراکنش‌های طولانی تجاری مانند بانک‌ها و فروشگاه‌های اینترنتی مناسب نیست، برای پیاده‌سازی یک تراکنش انعطاف‌پذیر با درخواست و پاسخ به دو چیز نیاز است:

- **Session**

  سرویس‌دهنده باید قادر به دریافت درخواست‌های ارسالی از یک سرویس‌گیرنده خاص و نگهداری آنها در یک session باشد. با تعریف session سیستم‌های بانکی و خریدهای اینترنتی قادر خواهند بود کاربران را از هم تشخیص دهند.

- **State**

  سرویس‌دهنده باید قادر به حفظ اطلاعات درخواست‌های قبلی و سایر اطلاعات تصمیم گیرنده به دست آمده از درخواست‌ها باشد. به عنوان مثال در یک فروشگاه تحت وب این اطلاعات می‌تواند شامل اقلام مورد علاقه کاربر، پروفایل کاربر و سبد خرید باشد.

به هرحال در پروتکل HTTP ارتباط در انتهای هر درخواست قطع می‌شود. بدین جهت سرویس‌دهنده‌های HTTP از ارتباط‌ها نمی‌توانند برای نگهداری session استفاده کنند و برنامه کاربردی قادر نیست تشخیص دهد که درخواست‌ها از سرویس‌گیرنده‌های مختلف یا از یک سرویس‌گیرنده خاص دریافت شده‌است.

در پروتکل HTTP یک تراکنش به چندین درخواست و پاسخ تقسیم می‌شود. سرویس‌دهنده وب نمی‌تواند تشخیص دهد همه درخواست‌ها از یک سرویس‌گیرنده ارسال شده‌است. بنابراین سرویس‌گیرنده نمی‌تواند برای انجام تراکنش‌های تجاری با سرویس‌دهنده گفتگو کند. هدف HTTP ایجاد یک ارتباط برای دریافت سریع و شفاف اطلاعات از طریق اینترنت است و این پروتکل ناپایدار به تنهایی برای پیاده‌سازی سیستم‌های تجاری کافی به نظر نمی‌رسد.

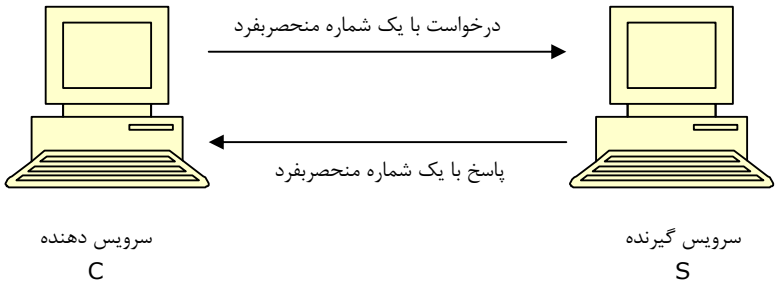
روش های ردیابی تماس کاربر

چهار مکانیزم اصلی برای ردیابی تماس[1] کاربر وجود دارد:

- دوباره نویسی URL
- کوکی‌ها[2]
- فیلدهای مخفی[3]
- Session با استفاده از SSL[4]

اگرچه پیاده‌سازی این چهار مکانیزم با هم متفاوت است ولی همه آنها به منظور سهولت در ردیابی تماس کاربر استفاده می‌شوند.

اگر سرویس‌گیرنده C و سرویس‌دهنده S درنظر گرفته شوند. هنگامیکه C برای اولین بار یک درخواست برای S ارسال می‌کند، S به C یک شماره منحصر بفرد می‌دهد. برای بار دوم که C یک درخواست برای S می‌فرستد، شماره منحصر بفرد خود را به همراه درخواست به S می‌فرستد و S می‌تواند C را توسط این شماره شناسایی کند. شکل زیر مکانیزم این چرخه را نمایش می‌دهد.



شکل 1-19 درخواست سرویس‌گیرنده و پاسخ سرویس‌دهنده

[1] Session Tracking
[2] Cookies
[3] Hidden Fields
[4] Secure Sockets Layers

این مکانیزم ساده برای ردیابی کاربر را می‌توان به روش‌های متفاوت پیاده‌سازی کرد.

- **دوباره‌نویسی URL**

در این روش شماره شناسایی کاربر به URL اضافه می‌شود، در هر صفحه پویایی که توسط سرویس‌دهنده ایجاد می‌شود به آدرس آن یک پارامتر اضافه می‌شود. هنگامیکه سرویس‌گیرنده درخواستی را با این آدرس به سرویس‌دهنده ارسال می‌کند، شماره شناسایی کاربر نیز ارسال خواهد شد. به این روش دوباره‌نویسی URL[1] گفته می‌شود.

- **فیلدهای مخفی**

این روش مشابه روش دوباره‌نویسی URL است، با این تفاوت که بجای دوباره‌نویسی URL سرویس‌دهنده اطلاعات شناسایی کاربر را در فیلدهای مخفی[2] موجود روی فرم‌های HTML قرار می‌دهد. هنگامیکه سرویس‌گیرنده فرم را ارسال می‌کند، فیلدهای مخفی حاوی اطلاعات شناسایی کاربر نیز همراه درخواست ارسال می‌شود. سرویس‌دهنده با استفاده از این اطلاعات می‌تواند کاربران را از هم تشخیص داده و ردیابی کند.

- **کوکی**

کوکی‌ها[3] اولین بار توسط شرکت Netscape مطرح شدند و یکی از بهترین روش‌های مبادله اطلاعات شناسایی کاربر بین سرویس‌گیرنده و سرویس‌دهنده می‌باشند. برخلاف دو روش قبلی، کوکی‌ها اطلاعات شناسایی را همراه HTTP Header درخواست و پاسخ ارسال می‌کنند، در نتیجه نیازی به دستکاری محتوای صفحات نیست.

- **Session با استفاده از SSL**

SSL یک تکنولوژی رمزنگاری در لایه بین TCP/IP و پروتکل‌های کاربردی مانند HTTP می‌باشد. SSL از پروتکل HTTPS استفاده می‌کند. SSL به سرویس‌دهنده‌های خود اجازه می‌دهد با سرویس‌گیرنده‌ها اطلاعات را در یک ارتباط رمزشده رد و بدل کنند. در طی برقراری یک ارتباط رمزشده، سرویس‌دهنده و سرویس‌گیرنده اطلاعاتی تحت عنوان "session keys" ایجاد می‌کنند که کلید رمزنگاری و از رمز در آوردن پیغام‌ها می‌باشد.

---

[1] URL Rewriting
[2] Hidden Fields
[3] Cookies

سرویس‌دهنده‌های پشتیبانی‌کننده HTTPS می‌توانند از این کلیدها برای ایجاد session استفاده کنند.

چگونه می‌توان از این راه‌حل‌ها برای ایجاد و مدیریت session در برنامه‌های تحت وب استفاده کرد؟ در سرولت حامل‌های وب[1] مسئول ایجاد و مدیریت session می‌باشند. سرولت به حامل‌های وب اجازه استفاده از روش‌های دوباره‌نویسی URL، کوکی‌ها و SSL را برای ردیابی session می‌دهد. انتخاب یکی از این چهار روش بستگی به قابلیت‌های سرویس‌دهنده و سرویس‌گیرنده دارد.

قبل از پرداختن به امکانات API سرولت برای ردیابی session، جزئیات سه روش دیگر بیان خواهد شد.

دوباره نویسی URL

همانطور که قبلاً گفته شد، درخواست HTTP شامل آدرس سرویس‌دهنده وب و پارامترهای درخواست بصورت زوج نام/مقدار می‌باشد. مثال:

```
http://www.myserver.com/Servlet/getSchedule;uid=joe?begPeriod=3&endPeriod=6
```

در مثال بالا عبارت www.myserver.com معرف سرویس‌دهنده، getSchedule;uid=joe آدرس فایل درخواستی و begPeriod=3 و endPeriod=6 پارامترهای درخواست می‌باشند. از این نوع آدرس‌دهی در ارسال فرم‌هایی استفاده می‌شود که دارای متد Get هستند. در روش دوباره‌نویسی URL اطلاعات شناسایی کاربر به همراه هر URL فرستاده می‌شود. در مثال زیر به هر یک از URLها پارامتری به نام uid با مقدار joe اضافه شده‌است.

```
http://www.myserver.com/servlet/getSchedule;uid=joe?begPeriod=3&endPeriod=6
```

روش دوباره‌نویسی URL فقط در صفحه‌هایی کاربرد دارد که بصورت پویا تولید می‌شوند. برای صفحات ثابت HTML نمی‌توان از این روش استفاده کرد، چراکه پارامترهای URL به ازای هر کاربر، یک مقدار متفاوت خواهند داشت.

---

[1] Web Containers

فیلدهای مخفی

در این روش اطلاعات شناسایی کاربر در داخل فرم‌های HTML و بصورت فیلدهای مخفی قرار می‌گیرد .

```
<INPUT type="hidden" name="uid" value="joe">
```

هنگامیکه درخواست فرستاده می‌شود، سرویس‌دهنده اطلاعات شناسایی را بصورت قسمتی از درخواست دریافت می‌کند. کاربرد این روش نیز مشابه روش قبلی فقط در صفحات پویا است و علاوه‌بر این درخواست سرویس‌گیرنده باید شامل یک فرم HTML با فیلدهای مخفی باشد.

کوکی

کوکی‌ها عمومی‌ترین روش برای نگهداری و ردیابی session یا اطلاعات شناسایی سرویس‌گیرنده است. کوکی توسط شرکت Netscape ارائه شد و بعدها به استاندارد RFC2109 تبدیل شد. اطلاعات بیشتر راجع به کوکی‌ها را می‌توان در سایت زیر پیداکرد:

```
http://www.netscape.com/newsref/std/cookie-spec.html
```

یک کوکی، تکه کوچکی از اطلاعات متنی است که از طرف سرویس‌دهنده برای سرویس‌گیرنده ارسال‌شده و در آنجا ذخیره می‌شود. این اطلاعات همراه با درخواست‌های بعدی سرویس‌گیرنده برای سرویس‌دهنده ارسال می‌گردد.

هر کوکی شامل زوج نام/مقدار با مشخصات اضافی است که بین سرویس‌دهنده و سرویس‌گیرنده رد و بدل می‌شود. سرویس‌دهنده وب کوکی را در قسمت HTTP Header پاسخ با دستور set-cookie به شکل زیر قرار می‌دهد:

```
Set-cookie :NAME=VALUE;Comment=COMMENT;
            Domain=DOMAINNAME;Max-age=SECONDS;
            Path=PATH;Secure;version=1*DIGIT
```

پارامتر NAME نام، VALUE مقدار و Max-age حداکثر مدت زمان (بر حسب ثانیه) اعتبار کوکی را مشخص می‌کند. Domain و Path اختیاری بوده و مشخص‌کننده آدرس و مسیری هستند که کوکی در آن معتبر است. اگر کوکی مجاز باشد از طریق HTTP ارسال شود کلمه Secure درج می‌شود. برای کوکی که براساس استاندارد RFC 2109 پیاده‌سازی شده‌است،

پارامتر version مقدار یک و برای کوکی که براساس مشخصات Netscape پیاده‌سازی شده است پارامتر version مقدار صفر خواهد داشت. پارامتر comment برای نوشتن توضیحات در کوکی استفاده می‌شود.

مثال:

```
Set-cookie:uid=joe;Max-age=3600;Domain=".myserver.com";path="/"
```

در مثال بالا یک کوکی به نام uid و مقدار joe با مدت زمان اعتبار 3600 ثانیه، برای سرویس‌دهنده myserver.com تعریف شده‌است.

سرویس‌گیرنده می‌تواند کوکی را رد و یا دریافت کند. به عنوان مثال اگر مرورگر طوری تنظیم شده‌باشد که کوکی را دریافت نکند، سرویس‌گیرنده کوکی را رد و در غیر اینصورت دریافت می‌کند.

هنگام ارسال درخواست بعدی، کوکی cookie:uid=joe به سرویس‌دهنده فرستاده می‌شود و بدین ترتیب سرویس‌دهنده قادر به تشخیص کاربران از یکدیگر خواهد شد. اگر سرویس‌گیرنده کوکی را رد کند، دیگر نمی‌تواند آن را برای سرویس‌دهنده ارسال کند و در نتیجه تلاش سرویس‌دهنده برای ردیابی کاربران به شکست می‌انجامد.

API سرولت یک کلاس به نام javax.servlet.http.Cookie ارائه می‌دهد که با استفاده از آن می‌توان یک کوکی ایجاد و سپس نام، مقدار و مدت زمان اعتبار آن را مشخص کرد و در نهایت کوکی را به شیء HttpServletResponse اضافه نمود. کوکی را می‌توان توسط شیء HttpServletRequest دریافت و مقدار آن را خواند. مثال:

```
// Create a new cookie with name and value arguments
  Cookie c = new Cookie("uid","joe");

  // Set the life of the cookie
  c.setMaxAge(60*60); //Expires in 1 hour
  c.setDomain(".myserver.com");
  c.setPath("/");

  // Send the cookie to the browser to be stored on the client machine
  response.addCookie(c);
```

ردیابی تماس کاربر در سرولت

در این API رابط javax.servlet.http.HttpSession برای ردیابی تماس کاربر تعریف شده‌است و حامل‌های وب این رابط را پیاده‌سازی می‌کنند.

از آنجاییکه یک session با درخواست کاربر شروع می‌شود، رابط HttpServletRequest متد getSession() را برای دسترسی به شیء HttpSession ایجاد شده در درخواست سرویس‌گیرنده، درنظر گرفته‌است. در دیاگرام javax.servlet.http بین رابطهای HttpServletRequest و HttpSession یک ارتباط وجود دارد که نشان می‌دهد برای دسترسی به شیء HttpSession از شیء HttpServletRequest استفاده می‌شود. رابط HttpSession متدهایی برای تنظیم و درک خصوصیات session ارائه داده‌است.

API سرولت برای ردیابی session از متدها و رابطهای زیر استفاده می‌کند:

- متدهای رابط javax.servlet.http.HttpServletRequest برای دسترسی به شیء HttpSession
- رابط javax.servlet.http.HttpSession برای نمایش session و متدهایی برای تعیین وضعیت و نامعتبر کردن session
- متدهای رابط javax.servlet.http.HttpServletResponse برای رمزنگاری URL در مواردی که مرورگر کوکی‌ها را رد کرده و از دوباره‌نویسی URL برای ردیابی session استفاده می‌کند.
- رابط javax.servlet.http.HttpSessionBindingListener برای پیگیری رویدادها.

ایجاد و ردیابی session

رابط HttpServletRequest متد getSession() را برای ایجاد و ردیابی session ارائه می‌دهد.

```
public HttpSession getSession(boolean create);
public HttpSession getSession();
```

این دو متد یک شیء HttpSession از درخواست جاری را برمی‌گردانند. اگر session برای درخواست جاری ایجاد نشده‌باشد، متد اول آن را ایجادکرده و متد دوم مقدار null برمی‌گرداند. از متد اول برای ایجاد و متد دوم برای گرفتن session استفاده می‌شود.

رابط HttpSession

این رابط از بسته javax.servlet.http مفهوم session را کپسوله می‌کند. یک شیء از نوع session با متد getSession() از شیء HttpServletRequest به دست می‌آید.

رابط HttpSession متدهای زیر را ارائه می‌دهد:

- public Object getAttribute(String name)
- public Enumeration getAttributeNames()
- public long getCreationTime()
- public String getId()
- public long getLastAccessedTime()
- public int getMaxInactiveInterval()
- public void invalidate()
- public boolean isNew()
- public void removeAttribute(String name)
- public void setAttribute(String name)
- public void setMaxInactiveInterval (int interval)

متدهای بالا را می‌توان به دو دسته تقسیم کرد:

1. متدهای طول حیات session

- public long getCreationTime()

این متد زمان ایجاد یک session را برحسب میلی ثانیه از تاریخ مبدا(ساعت صفر اول ژانویه سال 1970 به وقت گرینویچ) برمی‌گرداند.

- public long getLastAccessedTime()

این متد زمان آخرین دسترسی سرویس‌گیرنده به یک session را برحسب میلی ثانیه از تاریخ مبدا برمی‌گرداند. از این متد برای تعیین مدت زمان بین دو درخواست متوالی یک سرویس‌گیرنده نیز استفاده‌می‌شود.

- public int getMaxInactiveInterval()

این متد مدت زمانیکه session می‌تواند بین درخواست‌های سرویس‌گیرنده فعال بماند را برحسب ثانیه برمی‌گرداند.

- public void invalidate()

این متد برای باطل‌کردن session و آزادسازی منابع اختصاص داده‌شده استفاده می‌شود. بفرض قرار است بطور صریح session را با پیاده‌سازی متد logout یا logoff در برنامه باطل کرد. هنگامیکه کاربر این گزینه را انتخاب کند، با فراخوانی این متد سرویس‌دهنده session را نامعتبر کرده و کاربر دیگر نمی‌تواند از آن استفاده کند. برای ایجاد session جدید سرویس‌دهنده متد getSession() را از رابط HttpServletRequest فراخوانی می‌کند.

- public boolean isNew()

اگر session برای اولین بار توسط سرویس‌دهنده ایجاد شده باشد، این متد مقدار true و درغیراینصورت مقدار false برمی‌گرداند.

- public int setMaxInactiveInterval()

این متد مدت زمانیکه session می‌تواند بین درخواست‌ها قبل از باطل‌شدن، فعال بماند را تنظیم می‌کند. حامل‌های وب session را پس از سپری‌شدن زمان اعتبار باطل خواهند کرد. روش دیگر برای تنظیم این زمان، استفاده از تگ <session-timeout> در فایل XML پیکربندی سرویس‌دهنده وب است.

```
<session-config>
<session-timeout>300 </session-timeout>
</session-config>
```

2. متدهای ذخیره و بازیابی اطلاعات session

□ `public Object getAttribute(String name)`
این متد مقدار مشخصه معرفی‌شده با پارامتر name را بصورت یک Object برمی‌گرداند. اگر مشخصه موردنظر در session تعریف نشده‌باشد، مقدار null برمی‌گرداند.

□ `public Enumeration getAttributeNames(String name)`
این متد نام تمام مشخصه‌های تعریف شده در session را بصورت یک متغیر شمارشی برمی‌گرداند. با متد getAttribute() می‌توان به مقدار هر یک از این مشخصه‌ها دسترسی پیداکرد.

□ `public void setAttribute(String name, Object attribute)`
این متد یک مشخصه را با نام و مقدار آن در session جاری ذخیره می‌کند. این مشخصه با متد getAttribute() قابل دسترس می‌باشد.

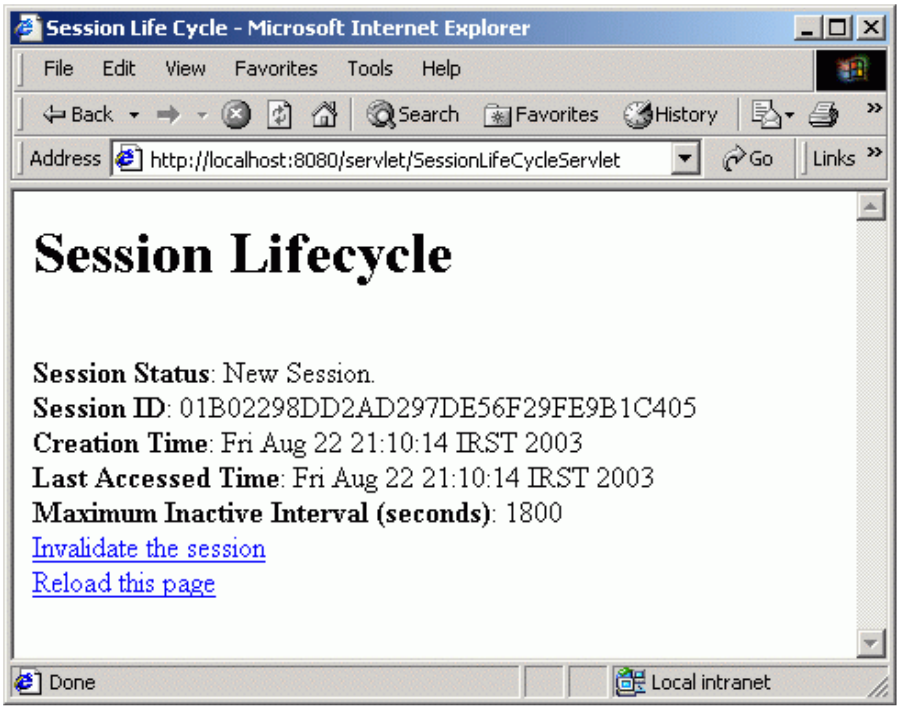
□ `public void removeAttribute(String name)`
این متد یک مشخصه را از session جاری حذف می‌کند.

متدهای setAttribute() و getAttribute() از رابط HttpSession امکان تعریف یکسری مشخصات (متغیر) را در شیء از نوع HttpSession فراهم می‌کنند، بطوریکه هر زمان قبل از باطل‌شدن session می‌توان آنها را دریافت کرد. هر مشخصه با نام آن تعریف می‌شود. قابل ذکر است متدهای فوق Synchronized نیستند.

نمایش چرخه حیات session توسط کوکی

درباره چرخه حیات شیء HttpSession در قسمت قبل بحث شد. حال وضعیت‌های مختلف session ایجادشده توسط یک سرولت و نحوه نامعتبر کردن آن مورد بحث و بررسی قرار می‌گیرد.

در ابتدا نحوه پیاده‌سازی session با استفاده از کوکی‌ها و سپس پیاده‌سازی session با استفاده از روش دوباره‌نویسی URL نمایش داده می‌شود. هنگام فراخوانی کلاس SessionLifeCycleServlet وضعیت، شماره منحصر بفرد session و نحوه ایجاد و باطل‌کردن session نمایش داده می‌شود. جهت انجام این کار از متدهای رابط HttpSession استفاده می‌شود. علاوه‌بر این نحوه فراخوانی مجدد صفحه و باطل‌کردن session نمایش داده می‌شود. شکل زیر خروجی مثال را نمایش می‌دهد.



شکل 2-19 خروجی اول برنامه SessionLifeCycleServlet

```
//Import Servlet packages
import javax.servlet.*;
import javax.servlet.http.*;

//Import Java packages
import java.io.*;
import java.util.Date;

public class SessionLifeCycleServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse
                         response) throws ServletException, IOException {

        String action = request.getParameter("action");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String lifeCycleURL = request.getServerName() + ":"
                            + request.getServerPort()
                            + "/servlet/SessionLifeCycleServlet";

        if(action != null && action.equals("invalidate")){
            HttpSession session = request.getSession();
            session.invalidate();
       else {
                HttpSession session = request.getSession();
                out.println("<HTML>");
                out.println("<HEAD><TITLE>Session Life Cycle</TITLE>");
                out.println("<META HTTP-EQUIV=\"Pragma\" CONTENT=\"no-cache\">");
                out.println("</HEAD>");
                out.println("<BODY bgcolor=\"#FFFFFF\">");
                out.println("<H1>Session Lifecycle</H1>");
                out.println("<BR><B>Session Status</B>: ");
                if(session.isNew()){
                    out.println("New Session.");
                } else {
                    out.println("Old Session.");
                }
                out.println("<BR><B>Session ID</B>: " + session.getId());
                out.println("<BR><B>Creation Time</B>: "
                                + new Date(session.getCreationTime()));
                out.println("<BR><B>Last Accessed Time</B>: "
```

کد 1-19 برنامه SessionLifeCycleServlet.java

بلاک else هنگامیکه سرولت بدون پارامتر فراخوانی شود، اجرا شده و مراحل زیر را طی می‌کند:

- یک شیء HttpSession با فراخوانی متد getSession() با پارامتر true ایجاد می‌کند.
- با فراخوانی متد getId() شماره منحصر بفرد session را نمایش می‌دهد.
- زمان ایجاد session را با فراخوانی متد getCreationTime() به یک شیء از نوع Date تبدیل می‌کند.
- آخرین زمان دسترسی به session را با فراخوانی متد getLastAccessedTime() نمایش می‌دهد.

- حداکثر زمان وقفه بین فراخوانی session را با فراخوانی متد getMaxInactiveInterval() نمایش می‌دهد.

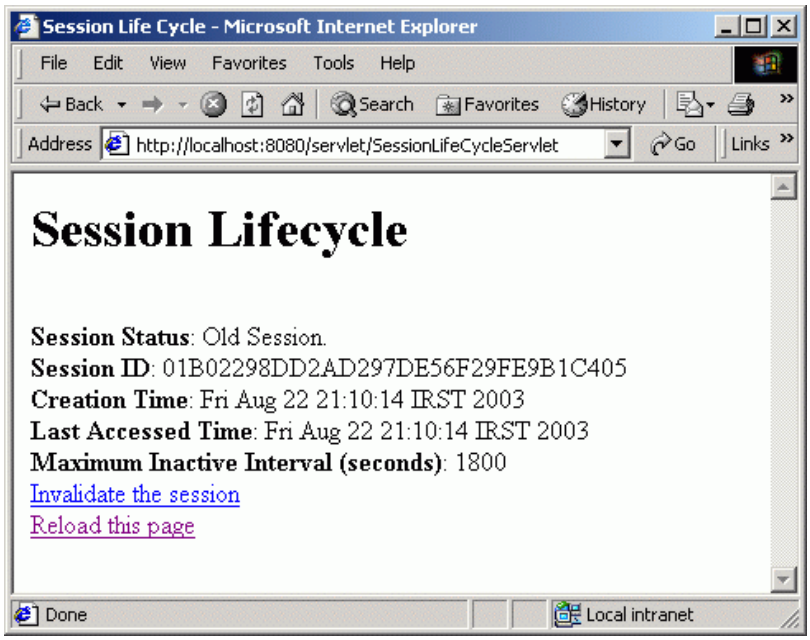
پس از چاپ اطلاعات بالا، دو گزینه برای فراخوانی مجدد صفحه و دیگری برای باطل‌کردن session چاپ می‌شود. مشخصه no-cache در HTML باعث دوباره بارگذاری صفحه در فراخوانی مجدد آن می‌شود.

گزینه دوم، پارامتر action=invalidate را به آدرس صفحه اضافه کرده و این بار بلاک if متد doGet() کلاس SessionLifeCycleServlet اجرا می‌شود.

برای اجرای این مثال ابتدا باید سرویس‌دهنده Tomcat را فعال کرده و سپس آدرس زیر را در مرورگر وارد کرد:

```
http://localhost:8080/session/servlet/SessionLifeCycleServlet
```

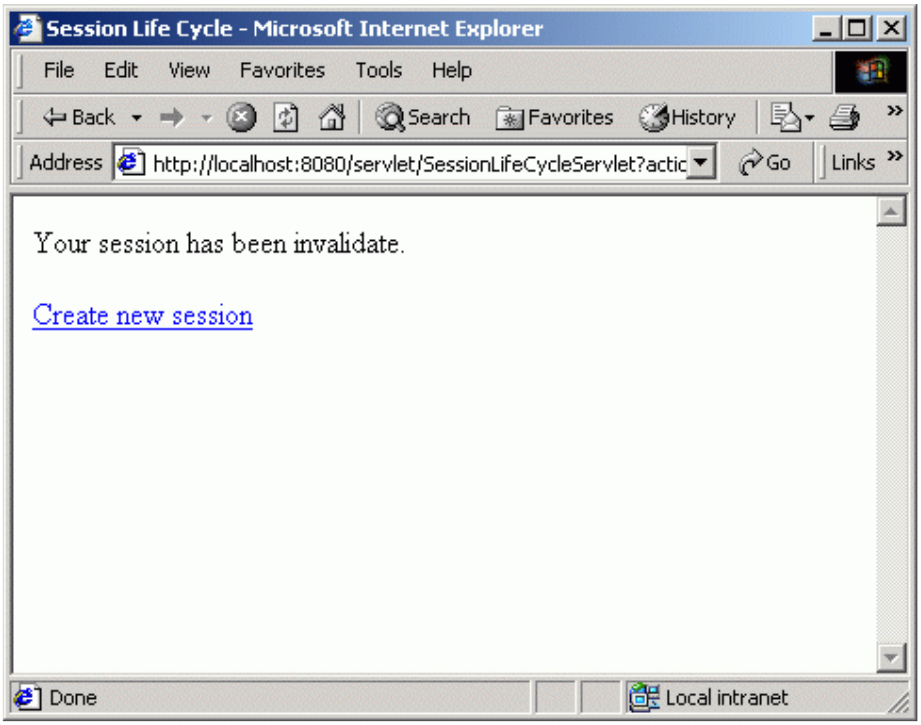
قبل از اجرا باید اطمینان حاصل کرد که مرورگر وب قادر به دریافت کوکی است. خروجی مثال بصورت زیر خواهد بود.



شکل 3-19 خروجی دوم برنامه SessionLifeCycleServlet

به ازای هر بار فراخوانی صفحه، شماره منحصر بفرد و زمان ایجاد session ثابت خواهد ماند ولی آخرین زمان دسترسی تغییر می‌یابد.

درصورت انتخاب گزینه Invalidate The Session، پارامتر action=invalidate به URL صفحه اضافه و session باطل و در نهایت خروجی بصورت زیر خواهد شد.



شکل 4-19 خروجی سوم برنامه SessionLifeCycleServlet

با انتخاب گزینه Create New Session بلاک else متد doGet() اجرا می‌شود.

چرخه حیات session بدون کوکی

برای مشاهده رفتار مثال SessionLifeCycleServlet بدون کوکی، ابتدا باید گزینه دریافت کوکی را در مرورگر غیر فعال کرده و سپس آدرس زیر را در آن واردکرد :

```
http://localhost:8080/session/servlet/SessionLifecycleServlet
```

به ازای هر بار فراخوانی صفحه، یک session جدید ایجاد می‌شود. بنابراین حامل وب نمی‌تواند session را ردیابی کند.

همانطور که قبلاً گفته شد اگر سرویس‌گیرنده کوکی را رد کند، حامل‌های وب از روش دوباره‌نویسی URL برای ردیابی session استفاده می‌کنند. بهرحال برای پیاده‌سازی این روش، URL باید با فراخوانی متد encodeURL() از رابط HttpServletResponse به رمز درآید. این متد شماره منحصر بفرد session را به انتهای URL اضافه خواهدکرد.

در مثال SessionLifeCycleServlet خط زیر را

```
String lifeCycleURL="/session/servlet/lifecycle";
```

باید با جمله زیر تغییر داد :

```
String lifeCycleURL = response.encodeURl("/session/servlet/lifecycle");
```

سپس باید کلاس را دوباره کامپایل و سرویس‌دهنده Tomcat را نیز دوباره اجرا کرد. این بار حامل وب قادر به ردیابی session می‌باشد و شماره session ثابت خواهد ماند. URL بصورت زیر درمی‌آید.

```
http://localhost:8080/session/servlet/SessionLifeCycleServlet;
                                        jsessionid=To1010mc89788362219823938
```

متدهای مدیریت وضعیت

همانطور که قبلاً گفته شد، یکی دیگر از نیازمندی‌های مهم برای ساخت برنامه‌های تحت وب علاوه بر شناسایی کاربر، بخاطر سپردن و نگهداری درخواست‌ها و اطلاعات تصمیم‌گیری کاربر است. به زبان ساده هنگام دریافت درخواست کاربر یا اتخاذ تصمیم جدید از طرف او باید امکان ذخیره‌سازی این اطلاعات جهت استفاده بعدی وجود داشته باشد.

به عنوان مثال، باتوجه به اطلاعات شناسایی کاربر، برنامه بانک اینترنتی اجازه برداشت سرمایه را از حساب خود نمی‌دهد. اگر برنامه قادر به ذخیره‌سازی این اطلاعات باشد، به ازای هر برداشت پول، نیازی به اجرای دوباره این تصمیم (منطق) نخواهدبود. می‌توان یک فرم ثبت نام را درنظر گرفت، اگر مراحل ثبت نام شامل حجم زیادی از اطلاعات (پرکردن چهار یا پنج فرم) باشد، ممکن است بجای یک صفحه با چند فرم، بهتر باشد اطلاعات را در چند صفحه با یک فرم ولی پشت سر هم دریافت کرد. برای انجام این کار باید قادر به تایید اعتبار و ذخیره اطلاعات تک تک فرم‌ها بود.

در ادامه، مثال دیگری برای نمایش متدهای مدیریت وضعیت مورد بررسی قرار می‌گیرد. کد زیر در فایلی به نام `AttributeServlet.java` ذخیره می‌گردد.

```
//Import Servlet packages
import javax.servlet.*;
import javax.servlet.http.*;

//Import Java packages
import java.io.*;
import java.util.Enumeration;

public class AttributeServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession();
        String name = request.getParameter("attrib_name");
        String value = request.getParameter("attrib_value");
        String remove = request.getParameter("attrib_remove");

        if(remove != null && remove.equals("on")){
            session.removeAttribute(name);
        } else {
            if(name != null && name.length() > 0
                    && value != null && value.length() > 0 ) {
                session.setAttribute(name,value);
            }
        }

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Session Attributes</TITLE>");
        out.println("<META HTTP-EQUIV=\"Pragma\"CONTENT=\"no-cache\">");
        out.println("</HEAD>");
        out.println("<BODY bgcolor=\"#FFFFFF\">");
        out.println("<H1>Session Attributes</H1>");
        out.println("Enter name and value of an attribute");
```

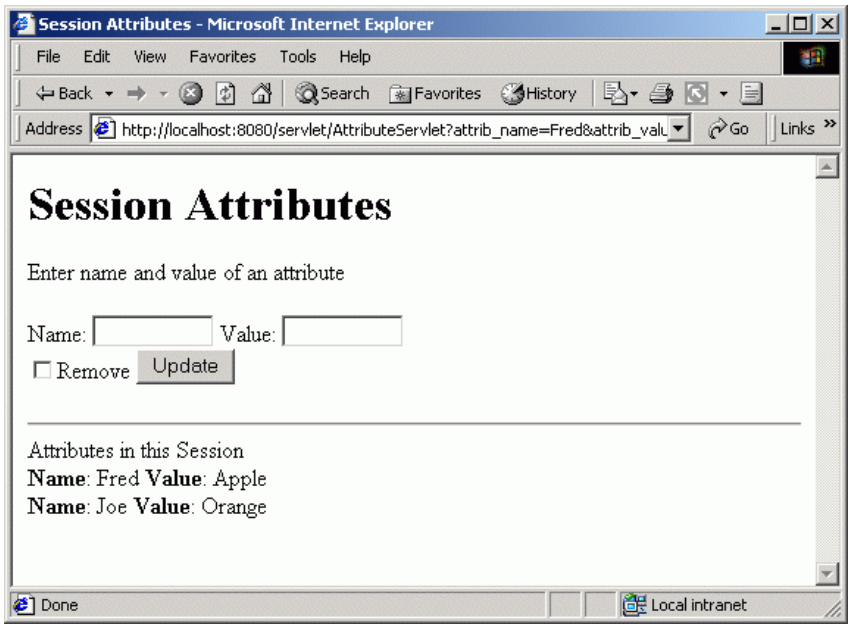
```
        String url = response.encodeURL("/servlet/AttributeServlet");

        out.println("<FORM action=\"" + url + "\"method=\"GET\">");
        out.println("Name: ");
        out.println("<INPUT type=\"text\" size=\"10\"name=\"attrib_name\">");
        out.println("Value: ");
        out.println("<INPUT type=\"text\" size=\"10\"name=\"attrib_value\">");
        out.println("<BR><INPUT type=\"checkbox\"
                            name=\"attrib_remove\">Remove");

        out.println("<INPUT type=\"submit\" name=\"update\" value=\"Update\">");
        out.println("</FORM>");
        out.println("<HR>");
        out.println("Attributes in this Session");

        //Print all session atrribute
        Enumeration e = session.getAttributeNames();
        while(e.hasMoreElements()){
            String att_name = (String) e.nextElement() ;
            String att_value = (String)session.getAttribute(att_name);
            out.println("<BR><B>Name</B>: ");
            out.println(att_name);
            out.println("<B>Value</B>: ");
            out.println(att_value);
        }
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

کـد 2-19 برنامه AttributeServlet.java

مثال بالا یک فرم HTML برای تعریف مشخصه با وارد کردن نام و مقدار در اختیار کاربر می‌گذارد. همچنین امکان حذف یک مشخصه از session وجود دارد. پس از کامپایل کردن مثال و قرار دادن در دایرکتوری web-inf سرویس‌دهنده وب Tomcat باید آدرس زیر را وارد کرد.

http://localhost:8080/servlet/AttributeServlet

خروجی آن در زیر مشاهده می‌گردد.



شکل 5-19 خروجی برنامه `AttributeServlet.java`

برای حذف هر مشخصه از `session` باید نام آن را در فیلد `Name` وارد و گزینه `Remove` را انتخاب و دکمه `update` را کلیک کرد.

این مثال ساده، چگونگی ذخیره و بازیابی مشخصه‌ها را در شیء HttpSession با استفاده از متدهای setAttribute()، getAttribute()، getAttributeNames() و removeAttribute() نشان داده است.

خلاصه

- HTTP یک پروتکل ناپایدار است که در آن ارتباط بین سرویس‌گیرنده و سرویس‌دهنده پس از ارسال یا دریافت اطلاعات قطع می گردد. ردیابی و پیگیری درخواست های یک کاربر یکی از نیازهای اصلی برنامه های مبتنی بر وب است.
- چهار مکانیزم اصلی برای ردیابی تماس کاربر وجود دارد: دوباره نویسی URL ، کوکی‌ها ، فیلدهای مخفی و Session با استفاده از SSL.
- رابط javax.servlet.http.HttpSession برای ردیابی تماس کاربر تعریف شده‌است و حامل‌های وب این رابط را پیاده‌سازی می‌کنند.
- این رابط شامل متدهایی برای مدیریت طول حیات session و ذخیره و بازیابی اطلاعات session است.