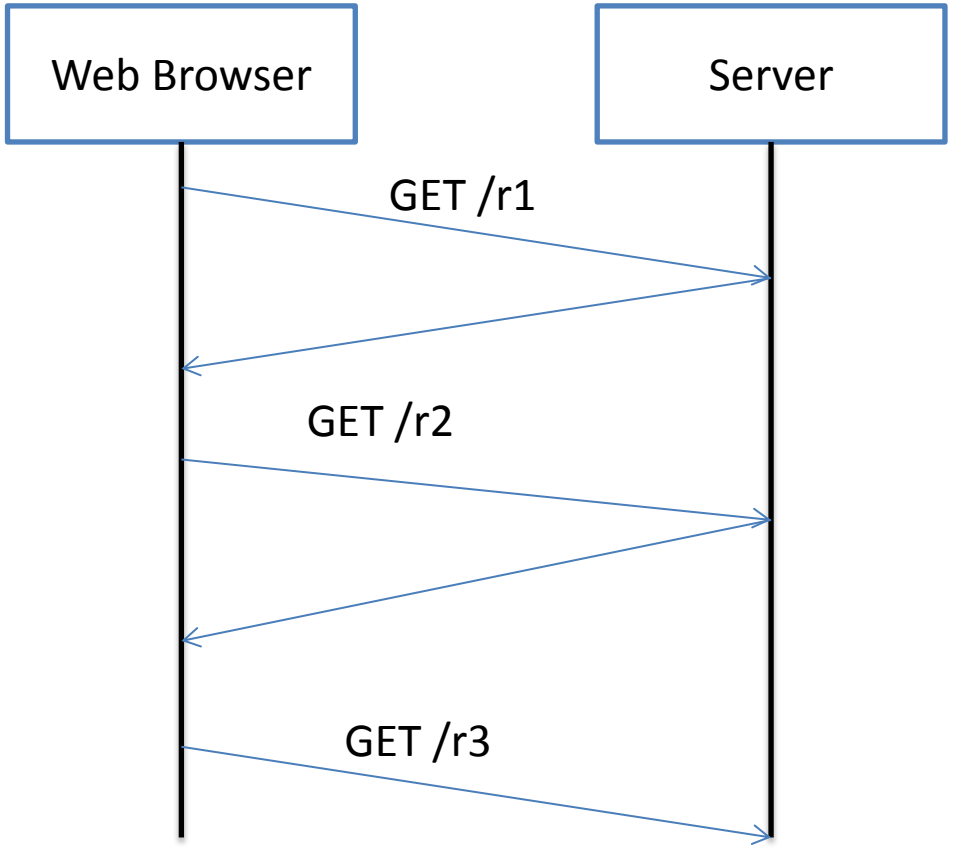
Java EE

Cookie et session

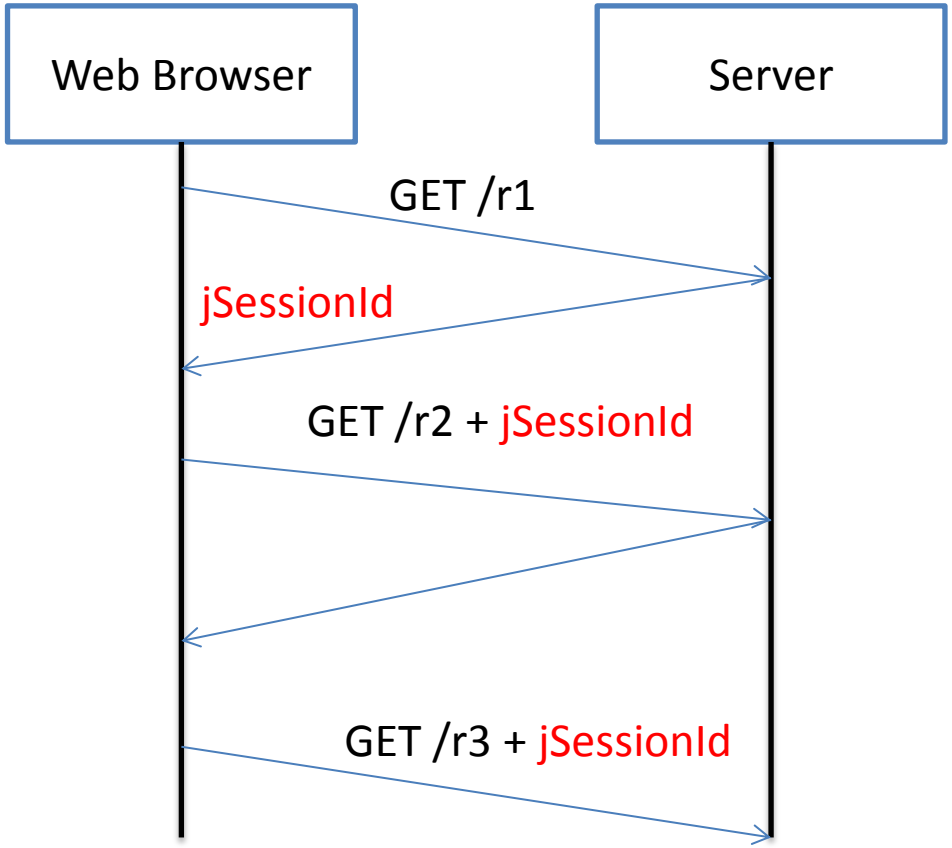
Le problème de suivi des sessions

- HTTP est un protocole sans état
- Le protocole ne fourni aucun moyen au serveur d'associer deux requêtes à un même client



Suivi des sessions en java

- Utilisation d'un jSessionId pour suivre les requêtes d'un client
- Le jSessionId est généralement stocké en tant que cookie par le bowser
- Si les cookies sont désactivés il peut être encodé dans l'URL



API servlet

L'API servlet offre une manipulation intuitive des session:

- Récupération d'une session auprès de l'objet HttpRequest
- Manipulation de la session à l'aide des méthodes
  - setAttribute (String name, Object o)
  - getAttribute (String name)
  - removeAttribute (String name)

```
// Récupération d'un attribut depuis la session
Object userObj = req.getSession.getAttribute("user");

if(userObj != null){
        User user = (User) userObj;
}
```

Durée de vie d'une session

- Une session peut expirer de plusieurs manières:
  - Si l'utilisateur ferme son navigateur
  - S'il reste inactif pendant plus d'une certaine durée

- La durée maximum d'inactivité d'une session peut se configurer à l'aide de la méthode setMaxInactiveInterval(int seconds) sur l'objet HttpSession

Manipulation d'un session

```
// Affichage de tous les attributs d'une session
Enumeration names = session.getAttributeNames();
while (names.hasMoreElements())
{
    System.out.println((String) names.nextElement());
}

// Récupération du jSessionId
String jSessionId = session.getId();

// Positionner la durée d'inactivité d'une session
session.setMaxInactiveInterval(60*60*24);  // une journée
session.setMaxInactiveInterval(-1); // jusqu'à la fermeture du browser

// Invalidation manuelle de la session
session.invalidate();
```

Session thread safe

- Un objet HttpSession est crée pour chaque client (ip + port)
- Avec une seule fenêtre (onglet) d'ouverte l'objet session est thread safe
- Cependant, en ouvrant plusieurs fenêtres, il est possible (bien que peu probable) que plusieurs thread accèdent en même temps au même objet session…

Session thread safe

Session thread safe

- Solution: utiliser le mot clé synchronize

```
Cart cart;
synchronized(session)
{
    cart = (Cart) session.getAttribute("cart");
}
```

Les cookies

- Un cookie est un stockage de type clé/valeur côté browser
- Les cookies sont crées par le serveur puis stocker par le browser
- A chaque requête le browser renvoie le cookie au serveur
- Les cookies peuvent avoir des durées de vie très longue (plusieurs mois)
- Les navigateurs possèdent des limitations sur le nombre et la taille des cookies autorisés par domaine et sous domaine

API servlet

- La manipulation d'un cookie est tout aussi simple qu'une session
- Constructeur: Cookie(String name, String value)
- Les object HttpRequest et HttpResponse proposent des méthodes getCookie(String name) et setCookie(Cookie c)

Example

```
// Création d'un cookie et ajout à la réponse
Cookie userIdCookie = new Cookie("userIdCookie", userId);
userIdCookie.setMaxAge(60*60*24*365*2); // 2ans !
userIdCookie.setPath("/"); // Autorisation sur toute l'application
response.addCookie(userIdCookie);

// Récupération d'un cookie
Cookie[] cookies = request.getCookies();
String cookieName = "userIdCookie";
String cookieValue = "";
for (int i=0; i<cookies.length; i++)
{
    Cookie cookie = cookies[i];
    if (cookieName.equals(cookie.getName()))
        cookieValue = cookie.getValue();
}
```

Cas d'utilisation des cookies

- Rendre une session utilisateur persistante (fonction « se souvenir de moi »)
- Personnaliser une page avec des widgets: cours de bourse, résultats de sport, météo…
- Améliorer l'efficacité des publicité
- Réduire le stockage côté serveur…