WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK

A **REST** (**Representational State Transfer**) egy szoftverarchitektúra típus, elosztott kapcsolat (loose coupling), nagy, internet alapú rendszerek számára, amilyen például a világháló.

A Representational State Transfer kifejezést Roy Fielding vezette be és definiálta 2000-ben a doktori disszertációjában. Fielding egyike a HTTP specifikáció szerkesztőinek.

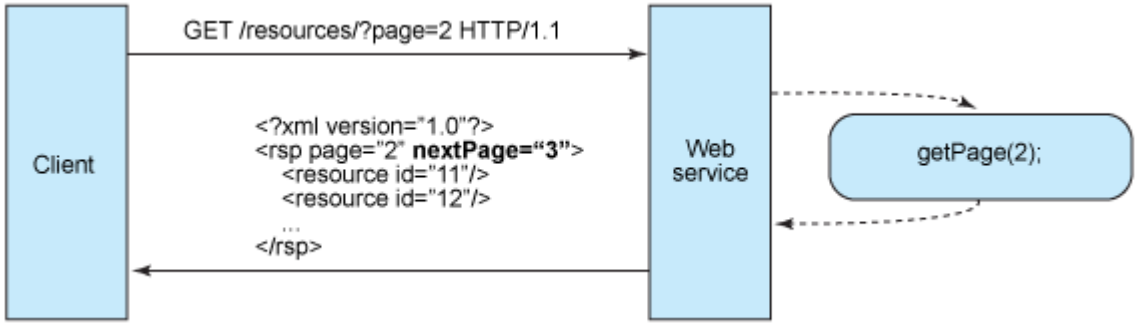
A RESTFUL webszolgáltatásoknak eleget kell tenniük néhány megszorításnak. Ezeket nézzük a következőkben.

WEB-programozás II (4. előadás)
RESTFUL WEBSZOLGÁLTATÁSOK

Használjuk a helyes HTTP metódus:

- Használjuk a **POST** metódust erőforrás létrehozására.
- Használjuk a **GET** metódust erőforrás lekérésére.
- Használjuk a **PUT** metódust az erőforrás vagy az állapotának a módosítására.
- Használjuk a **DELETE** metódust egy erőforrás törlésére, megszüntetésére.

Állapot nélküli kommunikáció:



WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK

Szerver:

- Generál más erőforrásokra hivatkozó linkeket tartalmazó válaszokat, hogy az alkalmazások navigálhassanak a címzett erőforrások között. Hasonlóképpen, ha a kérés egy szülő vagy egy konténer erőforrásra irányul, akkor egy tipikus RESTful válasz tartalmazhat linkeket a szülő gyermekeire vagy az alárendelt erőforrásokra annak érdekében, hogy azok továbbra is kapcsolatban maradjanak.
- Generál olyan válaszokat, amelyek azt jelzik, hogy cacheable vagy nem, a teljesítmény növelésének az érdekében, mert így csökkenthetjük a forgalmat. A szerver ezt úgy éri el, hogy visszaad egy Cache-Control és egy Last-Modified (dátum) paramétert a HTTP válasz fejlécében.

Kliens alkalmazás:

- A Cache-Control alapján eldönti, hogy csinál-e az erőforrásnak egy helyi másolatát. A kliens egy erőforrás lekérésekor a LastModified alapján küld egy úgynevezett feltételes GET kérelmet, amelynek a fejlécében van egy If-Modified-Since paraméter. Egy feltételes GET kérelemre kétféleképpen válaszolhat a szerver: visszaad egy szabványos 304 (Not Modified) kódot és kihagyja az erőforrást, ha az nem módosult az If-Modified-Since paraméterben kapott dátum óta, vagy visszaadja az erőforrást, ha módosult.
- A kliens küld teljes kérelmeket, amelyek más kérelmektől függetlenül szolgálhatók. Erre a kliens használja a HTTP fejléc paramétereit pontosan úgy, ahogyan szerepel a webszolgáltatás interfészében, és küldi a HTTP kérelem tartalmában az erőforrások teljes reprezentációit. A kliens olyan kérelmeket nyújt, amelyek nagyon keveset feltételeznek korábbi kérésekről, a szerveren létező munkamenetről (session-ról), a szerver képességéről, egy környezetben elhelyezné a kérelmet, vagy kérelmek közötti állapotokról.

Ez az együttműködés a kliens alkalmazás és a szolgáltatás között lényeges egy állapot nélküli RESTful webszolgáltatásban. Ez az együttműködés növeli a teljesítményt a sávszélesség megtakarításával és minimálizalva a szerver oldalon az alkalmazás állapotait.

WEB-programozás II (4. előadás)
RESTFUL WEBSZOLGÁLTATÁSOK

Használjuk könyvtár struktúra-szerű URI-kat:

Példák:

http://www.myservice.org/discussion/topics/{topic}
http://www.myservice.org/discussion/2008/12/10/{topic}
http://www.myservice.org/discussion/{year}/{day}/{month}/{topic}

- Rejtsük el a szerver-oldali szkriptfájlok kiterjesztéseit (.jsp, .php, .asp), ha van ilyen, így lehet váltani más technológiára az URI-k megváltoztatása nélkül.
- Használjuk mindig és mindenhol kisbetűket.
- Helyettesítsük a szóközöket kötőjelekkel vagy aláhúzásokkal (vagy az egyikkel, vagy a másikkal, de ne vegyesen).
- Kerüljük el, amennyire csak tudunk, a kérdő karaktersorozatokat.
- Használjuk egy alapfeltételezett oldalt vagy erőforrást válaszként a 404 Not Found kód helyett.

WEB-programozás II (4. előadás)
RESTFUL WEBSZOLGÁLTATÁSOK

Adjuk át XML-t, JSON-t, vagy mind a kettőt:

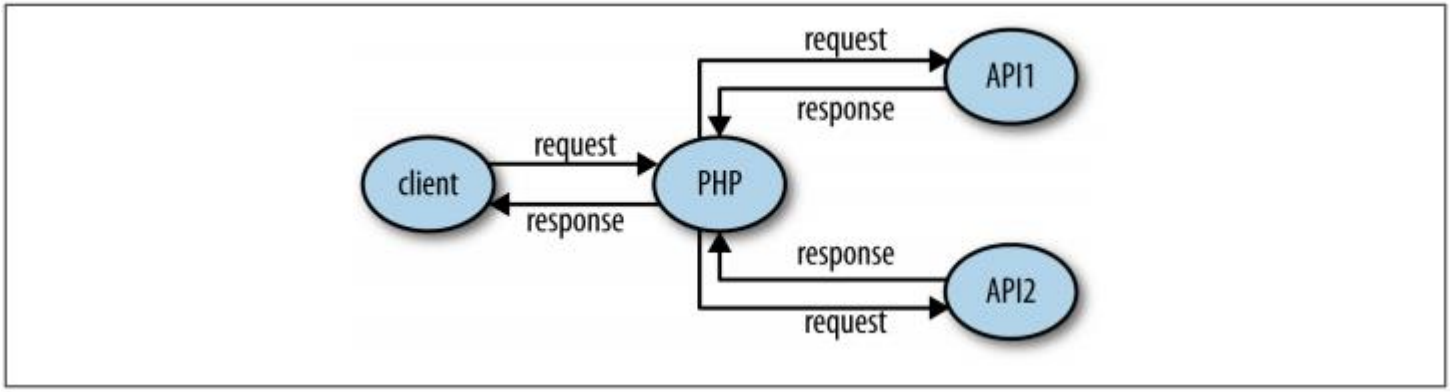
XML példa:

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
    <comment>{comment}</comment>
    <replies>
     <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe"/>
     <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob"/>
    </replies>
</discussion>
```

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

PHP alapú webalkalmazás, amely szerver szerepet tölt be a felhasználó szempontjából, de más API-k kliense:



Amennyiben az API-k RESTFUL webszolgáltatásokat valósítják meg, akkor a PHP alapú alkalmazásnak generálnia kell GET, POST, DELETE és PUT HTTP kérelmeket.

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

GET HTTP kérelem a CURL kiterjesztés segítségével:

- CURL kiterjesztés: a cURL (client URL) protokollfüggetlen fájlátviteli parancssoros könyvtár megvalósítása PHP-ben -

```
<?php
$url = "http://localhost/web2/";
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);

// mentsük meg és írjuk ki a választ
file_put_contents("get_curl.html", $result);
print_r($result);
?>
```

Megjegyzés:
A **curl_setopt($ch, CURLOPT_RETURNTRANSFER, true)** opció beállításával a **curl_exec($ch)** a kérelem válaszát adja vissza a kiíratása (**echo**-val) helyett.

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

GET HTTP kérelem a PHP stream kezelésének a segítségével:

```
<?php
$url = "http://localhost/web2/";
$result = file_get_contents($url);

// mentsük meg és írjuk ki a választ
file_put_contents("get_stream.html", $result);
print_r($result);
?>
```

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

POST HTTP kérelem a CURL kiterjesztés segítségével (1. változat):

```
<?php
$url = "http://localhost/web2/beleptet";
$data = "login=Login9&password=login9";
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $data);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);

// mentsük meg és írjuk ki a választ
file_put_contents("post_curl.html", $result);
print_r($result);
?>
```

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

POST HTTP kérelem a CURL kiterjesztés segítségével (2. változat):

```
<?php
$url = "http://localhost/web2/beleptet";
$data = array("login" => "Login9", "password" => "login9");
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);

// mentsük meg és írjuk ki a választ
file_put_contents("post_curl.html", $result);
print_r($result);
?>
```

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

POST HTTP kérelem a PHP stream kezelésének a segítségével:

```
<?php
$url = "http://localhost/web2/beleptet";
$data = array("login" => "Login9", "password" => "login9");
$context = stream_context_create(
    array(
        'http' => array(
                'method' => 'POST',
                'header' =>
                        array('Content-Type: application/x-www-form-urlencoded'),
                'content' => http_build_query($data)
         )
    )
);
$result = file_get_contents($url, false, $context);

// mentsük meg és írjuk ki a választ
file_put_contents("post_stream.html", $result);
print_r($result);
?>
```

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

DELETE HTTP kérelem a CURL kiterjesztés segítségével:

```
<?php
$url = 'http://localhost/book/example-delete.php';
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");
curl_exec($ch);
?>
```

A kiszolgáló PHP szkriptben megtudhatjuk a kapott kérelem metódusát a

$_SERVER["REQUEST_METHOD"]

változó ellenőrzésével, amely jelzi, hogy melyik metódust használ a kérelem.

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

PUT HTTP kérelem a CURL kiterjesztés segítségével:

```
<?php
$url = 'http://localhost/ea4/put_szerver.php';
$data = array("user" => "Xyz", "email" => "xyz@xyz.hu" );
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$result = curl_exec($ch);
curl_close($ch);

// mentsük meg és írjuk ki a választ
file_put_contents("put_curl_kliens.txt", $result);
print_r($result);
?>
```

WEB-programozás II (4. előadás)

RESTFUL WEBSZOLGÁLTATÁSOK PHP-BEN

Kérdés: Hogyan lehet feldolgozni egy ilyen kérelmet egy PHP szkript-ben, mert a $_POST globális változó létezik, de $_PUT nincs.

PUT HTTP kérelem feldolgozása PHP-ben:

```
<?php
if($_SERVER['REQUEST_METHOD'] == "PUT") {
    $data = array();
    $incoming = file_get_contents("php://input");
    parse_str($incoming, $data);
    echo "Felhasználó: ".$data["user"]." új email címe: " .
            filter_var($data["email"], FILTER_VALIDATE_EMAIL);
}
else {
    echo "um?";
}
?>
```

WEB-programozás II (4. előadás)

AUTENTIKÁCIÓ A HTTP-BEN

HTTP alap autentikáció:

Az egyik legegyszerűbb módja annak, hogy egy weboldalt biztosítsunk, a HTTP alap autentikációjának a használata. Ez azt jelenti, hogy elküldjük kódoltan a kliens autentikációját szolgáló adatokat az **Authorization** fejlécben minden HTTP kérelemben.

Ennek a megközelítésnek az alapmechanizmusa egyszerű: a kliensek rendelkeznek egy felhasználónévvel és egy jelszóval, amelyekkel a következőképpen járnak el:

1. Rendezik a felhasználónév-jelszó párt a felhasználónév:jelszó formátumban.
2. Base64 kódolják az eredményt.
3. Elküldik a fejlécben: Authorization: Basic base64-encoded string.
4. Használják HTTPS-t.

WEB-programozás II (4. előadás)

AUTENTIKÁCIÓ A HTTP-BEN

HTTP alap autentikáció PHP-ben a CURL kiterjesztés segítségével:

```
<?php
$url = "https://localhost/ea4/basic-auth.php";
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC ) ;
curl_setopt($ch, CURLOPT_USERPWD, "user:pass");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
echo $response;
curl_close($ch);
?>
```

WEB-programozás II (4. előadás)
AUTENTIKÁCIÓ A HTTP-BEN

HTTP alap autentikáció PHP-ben (szerver oldal):

A PHP-ben a **$_SERVER** változóban találhatók a felhasználó által megadott felhasználónév és jelszó:

$_SERVER["PHP_AUTH_USER"]

$_SERVER["PHP_AUTH_PW"]

Ha az autentikációt szolgáló adatok érvénytelenek vagy hiányosak, akkor a szerver egy **401 Unauthorized status code** küldhet a kliens számára, jelezve miért nem küldi el a kért tartalmat.

WEB-programozás II (4. előadás)

SOAP VS RESTFUL

A SOAP alapú webszolgáltatások műveleteket valósítanak meg. A SOAP minden bizonnyal a nehézsúlyú választás a webszolgáltatások megvalósításához.

A SOAP előnyei a REST-tel szemben:

- Nyelv, platform és kommunikációs protokoll független (a REST-hez a HTTP használata szükséges).
- Jól működik az elosztott vállalati környezetekben (a REST közvetlen point-to-point kommunikációt feltételez).
- Szabványosított.
- Jelentős pre-build kiterjeszthetőséget biztosít a WS * szabványok formájában.
- Beépített hibakezelés.
- Automatizálás bizonyos nyelvi termékek esetén.

WEB-programozás II (4. előadás)

SOAP VS RESTFUL

A REST erőforrásokkal dolgozik. A REST a legtöbb esetben könnyebben használható és rugalmasabb.

A REST a SOAP-hoz képest a következő előnyökkel rendelkezik:

- Nincs szükség drága eszközökre ahhoz, hogy kölcsönhatásba lépjenek , hogy használhassuk a webszolgáltatásokat.
- Kisebb tanulási idő.
- Hatékony (a SOAP minden üzenethez XML-t használ, a REST kisebb üzenetformátumokat használhat).
- Gyors (nem szükséges kiterjedt feldolgozás).
- Közelebb van más webes technológiákhoz a tervezési filozófiájában.
- A REST esetén az erőforrások tárolhatók cache memóriában, ezzel szemben a SOAP mindig POST metódust használ.

WEB-programozás II (4. előadás)

SOAP VS RESTFUL

Egyes források szerint a webszolgáltatások 70%-a REST alapú. A publikus webszolgáltatások körében a REST architektúra az elterjedtebb.

Néhány példa terület, ahol ajánlott SOAP alapú szolgáltatások használni a nagyobb megbízhatóságának köszönhetően:

- Pénzügy.
- E-kereskedelem.
- Vállalati környezet.