Webtechnológiák információs rendszerekben könyvfeldolgozás

Sándor Balázs – AZA6NL

A feldolgozott könyvfejezetek az alábbi könyvből származnak:

SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST (2013, Thomas Erl)

I.) Szolgáltatás-orientált tervezés REST segítségével

(173-227 Service-Oriented Design with REST)

A szolgáltatás-orientált tervezés a szolgáltatások fizikai tervezésére összpontosítja a szolgáltatás-csoportokat megelőző koncepcionális szolgáltatás-jelöltjeik modelljének felhasználásával. Amikor a REST szerződéstervezéssel foglalkozunk, két konkrét területre kell összpontosítanunk:

Egy egységes szerződés tervezése egy szolgáltatási leltárban.

Az egyedi szolgáltatási szerződések tervezése a szolgáltatási leltárban, és az egységes szerződés szabályozása szerint.

Az egységes szerződést szilárdan meg kell határozni, mielőtt olyan szolgáltatási szerződéseket kezdenénk létrehozni, amelyek függőséget fognak kialakítani az egységes szerződés jellemzőire. Ahogy egy szolgáltatási leltár nő és fejlődik, az új szolgáltatások még mindig befolyásolhatják az egységes szerződés tervezését, de az egységes szerződés jellemzőit általában nagyon szándékos ütemben változtatják és bővítik.

A korábbi sorrendet követve ez a fejezet az egységes szerződés tervezési témáinak áttekintésével kezdődik, majd áttér a REST szolgáltatási szerződések tervezéséhez kapcsolódó témákra. A fejezet az összetett módszerekkel zárul, amelyek opcionális területet képeznek a REST szerződéstervezésben, és amelyeket főként ellenőrzött környezetekben, például belső szolgáltatási leltárakban lehet használni.

10.1 Egységes szerződés tervezési megfontolások

Amikor egy egységes szerződést hozunk létre egy szolgáltatási leltárhoz, felelősségünk, hogy annak jellemzőit oly módon alakítsuk ki és korlátozzuk, hogy hatékonyan szolgálja a leltár egyedi követelményeit és korlátait. A Web-centric technológia alapvető jellemzői jó kiindulási pontot nyújtanak, bár a nem triviális leltárarchitektúrákhoz valószínűleg további standardizálás és testreszabás szükséges. A továbbiakban részletezem, hogyan lehet egy egységes szerződést testreszabni az egyedi szolgáltatási leltárak igényeinek kielégítésére.

Módszerek tervezése és szabványosítása

Az egységes szerződés kialakításánál fontos, hogy a módszereket úgy tervezzük és szabványosítsuk, hogy hatékonyan szolgálják a szolgáltatási leltár egyedi követelményeit és korlátozásait. Bár az alapvető módszereknek kell lenniük, új módszereket is hozzá lehet adni az igényeknek megfelelően. Az HTTP egy jó alapot biztosít, de más típusú interakciókat is kifejezhetünk. Fontos azonban, hogy óvatosan kezeljük az új kiterjesztéseket, és csak akkor alkalmazzuk őket, ha a környezet technológiai hatásait teljes mértékben megértettük.

HTTP fejléc tervezése és szabványosítása

Az HTTP fejléc tervezése és szabványosítása során fontos szerepet játszik az üzenetek cseréjével kapcsolatos metaadatok kezelése. Az alapbeépített HTTP fejléceknek különböző módon lehet felhasználásuk:

Paraméterek hozzáadására a kérési módszerhez, ami alternatívát jelenthet az URL-ben szereplő paraméterek használatához.

Paraméterek hozzáadására a válaszkódhoz, például az újonnan létrehozott erőforrás azonosítójának megadása.

Általános információk közlésére a szolgáltatás vagy a fogyasztó kapcsán.

Az egyedi HTTP fejlécek segítségével gazdagabb metaadatokat adhatunk át. Fontos megjegyezni, hogy az új módszerek bevezetésekor figyelembe kell venni, hogy az üzenet tartalma kötelezően értelmezendő-e vagy opcionálisan figyelmen kívül hagyható-e. Az ilyen fejlécek biztonságosan használhatók a szokásos HTTP módszerekkel, és visszafelé kompatibilisek lehetnek az üzenettípusok új verzióinak létrehozásakor. Fontolóra vehető az is, hogy egy új Must-Understand fejléc létrehozása, amely a szükséges érthető fejléceket tartalmazza, és a SOA irányítási programiroda felelőssége lenne az ilyen jellegű szemantikák egységes és konzisztens végrehajtása.

Http válaszkódok tervezése és szabványosítása

Az HTTP válaszkódok tervezése és szabványosítása kulcsfontosságú a szolgáltatásalapú megoldások tervezésekor. Az HTTP eredetileg szinkron, ügyfél-szerver protokollként lett kialakítva a HTML oldalak cseréjére a világhálón. Ezek a jellemzők kompatibilisek a REST korlátokkal, ezáltal alkalmassá teszik a protokollt a REST szolgáltatás képességeinek meghívására. Az HTTP szolgáltatás kialakítása hasonló a dinamikus tartalom közzétételéhez egy webkiszolgálón. Az egyes HTTP kérések REST szolgáltatásképességeket hívnak meg, és ezt követően válaszüzenet küldése történik a szolgáltatásfogyasztónak. A válaszüzenetekben számos HTTP kód lehet, melyek mindegyike kijelölt számmal rendelkezik, és bizonyos kódszám-tartományok különböző típusú feltételeket jelölnek. A különböző kódtartományok jelzik az információk különböző típusait, például a sikeres vagy sikertelen kéréseket, valamint az átirányításokat és hibákat. A válaszkódok standardizáltak, de az, hogy a szolgáltatásfogyasztók hogyan reagálnak ezekre, az nem. Ezért szükséges az egyértelmű válaszkódok értelmezését és kezelését meghatározó együttes szabálykészlet létrehozása a szolgáltatásalapú tervezés során. Az HTTP kódok további jelentéseket hordoznak, és lehetőséget adnak a fogyasztók számára a különböző típusú kivételek kezelésére és helyreállásra.

Pár szó az intervallumokról:

Az ***100-199*** tartomány a tájékoztató kódok, melyek alacsony szintű jelzési mechanizmusokként szolgálnak, például megerősítik a protokollváltás kérését.

A ***200-299*** tartomány általános siker kódok, különböző sikerkörülményeket írnak le.

A ***300-399*** tartomány átirányítási kódok, melyek arra szolgálnak, hogy a fogyasztó újra próbálkozzon egy másik erőforrásazonosító vagy egy másik köztesen keresztül.

A ***400-499*** tartomány a fogyasztó oldali hibakódok, melyek arra utalnak, hogy a fogyasztó valamilyen okból érvénytelen kérést küldött.

A ***500-599*** tartomány a szolgáltató oldali hibakódok, amelyek arra utalnak, hogy a fogyasztó kérése lehet érvényes, de a szolgáltatás belső okokból nem képes azt feldolgozni.

A fogyasztói és szolgáltatói oldalon lévő kivételes kategóriák segítenek a "felelősség megállapításában", de kevésbé járulnak hozzá a szolgáltatásfogyasztók valódi helyreállításához. Ez azért van, mert bár az HTTP által biztosított kódok és okok szabványosak, az üzenetkódok fogadása utáni viselkedésre vonatkozó elvárások nem. Amikor egy szolgáltatás-inventárium szolgáltatás tervezésének szabványosítása, szükséges egy olyan együttes szabálykészlet létrehozása, amely konkrét jelentést és kezelést rendel a válaszkódokhoz.

- ***Repeat*** - Ismételés azt jelenti, hogy a fogyasztónak ösztönözni kell a kérés ismétlését, figyelembe véve a válaszokban megadott esetleges késleltetéseket, például a 503 Szolgáltatás nem elérhető-t. Ez azt jelentheti, hogy várni kell, mielőtt újra megpróbálkozna. Ha a fogyasztó úgy dönt, hogy nem ismétli meg a kérést, annak kezelését sikertelennek kell tekinteni.

- ***Success*** - Sikeres azt jelenti, hogy a fogyasztónak sikeres műveletként kell értelmeznie az üzenet továbbítását, és ezért nem szabad megismételnie azt. (Fontos megjegyezni, hogy a konkrét sikerkódoknak finomabb értelmezése is lehet.)

- ***Failed*** - Sikertelen azt jelenti, hogy a fogyasztónak nem szabad változatlanul megismételnie a kérést, bár egy új kérést kiadhat, figyelembe véve a választ. Ha az új kérés nem generálható, a fogyasztónak a kérés sikertelenként kell kezelnie. (Fontos megjegyezni, hogy a konkrét hibakódoknak finomabb értelmezése is lehet.)

- ***Indeterminate*** - Bizonytalan azt jelenti, hogy a fogyasztónak módosítania kell a kérését a megadott módon. A kérést nem szabad változatlanul megismételni, és egy új kérést kell kiadni, figyelembe véve a választ. A kölcsönhatás végső kimenetele az új kéréstől függ. Ha a fogyasztó nem képes új kérést generálni, akkor ezt a kódot sikertelenként kell kezelni.

Mivel az HTTP egy protokoll, nem egy üzenetfeldolgozási logika halmaza, a szolgáltatásnak kell eldöntenie, hogy milyen státuszkódot (siker, hiba vagy egyéb) küldjön vissza. Mint korábban említettük, mivel a fogyasztói viselkedés nem mindig eléggé szabványos az HTTP által gép-gép közötti interakciókhoz, ezért az SOA projekt részeként egyértelműen és jelentés szerinti módon szabványosítani kell. Például az indeterminált kódok általában azt jelzik, hogy a szolgáltatásfogyasztóknak saját egyedi logikával kell kezelniük egy helyzetet. Ezeket a típusú kódokat két módon szabványosíthatjuk:

- A tervezési szabványok meghatározhatják, hogy melyik indeterminált kódokat lehet és melyeket nem lehet kibocsátani a szolgáltatás logikájából.

• A tervezési szabványok meghatározhatják, hogy a szolgáltatásfogyasztói logikának hogyan kell értelmeznie azokat az indeterminált kódokat, amelyeket engedélyeznek.

Válaszkódok testre szabása

Az HTTP specifikáció lehetővé teszi a válaszkódok testre szabását. Ez főként az új HTTP verziók bevezetésére szolgál, és más specifikációkhoz, például a WebDAV-hez is alkalmazható. A testreszabott kódokat érdemes numerikus tartományokba csoportosítani, és humán-olvasmányos tartalmat is hozzá lehet adni az HTTP válaszhoz. Fontos, hogy az egyedi kódokat megfelelően és körültekintően tervezzék, és ne ütközzenek más kódokkal vagy protokollspecifikációkkal. A válaszkódok tartományai kivétel-öröklődésként működnek, és a szolgáltatási készletek közötti üzenetcseréket is támogatják.

Média típusok tervezése

Média típusok tervezése során gyakran változnak a uniform szerződés média típusai. Például új média típusra lehet szükség, ha gépelhető információkat kell közölni, ami nem felel meg a meglévő típusoknak. Néhány közös média típus közé tartozik a text/plain, application/xhtml+xml, text/uri-list és az application/atom+xml. Ajánlott meglévő ipari média típusokat keresni mielőtt újat találunk ki. Legyen szó meglévő vagy egyedi típusokról, fontos a legjobb gyakorlatok figyelembevétele.

Minden média típusnak ideális esetben egy sémához kellene kapcsolódnia. Például az application/xml vagy az application/json nem specifikus sémához, míg az application/atom+xml elég specifikus ahhoz, hogy hasznos legyen a tartalomtárgyalásban és a dokumentumok feldolgozásában.

A média típusoknak absztraktaknak kell lenniük, csak annyi információt kell megadniuk, amennyire a címzetteknek a sémáikon keresztül szükségük van. Ez lehetővé teszi az újrafelhasználást több szolgáltatási szerződésen belül.

Az új média típusoknak érett szótárakat és fogalmakat kell újrafelhasználni, hogy csökkentsék a kockázatot és javítsák a kompatibilitást más alkalmazásokkal.

A média típusoknak tartalmazniuk kell hivatkozást, ha egy külső erőforrásra kell hivatkozniuk. Az egyedi média típusoknak kiterjesztési pontokkal kell rendelkezniük az új adatok hozzáadásához a későbbi verziókhoz.

Minden média típusnak standard feldolgozási utasításokat kell tartalmaznia az elavult dokumentumok kezeléséhez. Ezek lehetővé teszik az új szolgáltatásoknak és fogyasztóknak, hogy kompatibilisek legyenek az előző verziókkal.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://example.com/schema/po"
  xmlns="http://example.com/schema/po">
  <xsd:element name="LineItemList" type="LineItemListType"/>
  <xsd:complexType name="LineItemListType">
    <xsd:element name="LineItem" type="LineItemType"
      minOccurs="0"/>
  </xsd:complexType>
  <xsd:complexType name="LineItemType">
    <xsd:sequence>
      <xsd:element name="productID" type="xsd:anyURI"/>
      <xsd:element name="productName" type="xsd:string"/>
      <xsd:element name="available" type="xsd:boolean"
        minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Az összes új vagy újrafelhasznált média típust dokumentálni kell a uniform szerződés profiljában. Az egyedi média típusokat az általános jelölés szerint azonosítják, és opcionálisan regisztrálhatók az IANA-nál.

Média típusokhoz sémák tervezése

Média típusokhoz tervezett sémák legyenek rugalmasak és együttműködők. Az egyenletes szerződés szintjén definiált média típusokat és sémákat kell használni. Szolgáltatásspecifikus sémák bevezetése esetén figyelembe kell venni az Egyenletes Szerződés {400} megszorítását. A szolgáltatásspecifikus média típusoknak egyértelműen azonosítottnak kell lenniük, és minimalizálni kell a függő logikát.

Fő pontok összefoglalása

• Kialakíthatunk és szabványosíthatunk egyedi HTTP módszereket és válaszkódokat. Emellett standardizálhatjuk, hogy a beépített HTTP módszereket és válaszkódokat hogyan használják (vagy hogy használják-e).

• Számos meglévő média típus közül választhatunk (és újrafelhasználhatunk) egy szolgáltatási készleten belül, amelyek közül sok regisztrálva van az IANA-nál (és más iparági testületeknél). Emellett tervezhetünk és szabványosíthatunk egyedi média típusokat azáltal, hogy közös adat- és dokumentumtípusokat képviselünk, amelyeket a szolgáltatási készleten belül cserélnek ki.

• Azok a sémák, amelyeket média típusok foglalnak magukban, természetesen szabványosítódnak, amikor részükké válnak egy egyenletes szerződésnek. Ahhoz, hogy a sémák újrafelhasználhatók legyenek, általában rugalmasan kell tervezni őket, biztosítva a validációs korlátozások szintjének csökkentését.

10.2 REST szolgáltatási szerződés tervezése

Ebben a következő szakaszban olyan tervezési technikákat és szempontokat vizsgálunk, amelyek specifikusak az egyes REST szolgáltatási szerződésekhez, és hogy ezek hogyan kapcsolódnak az átfogó egyenletes szerződésükhöz.

Szolgáltatások tervezése szolgáltatási modelleken alapulva

Az adott REST szolgáltatás választása befolyásolhatja a szolgáltatási szerződés tervezési megközelítésünket. Az alábbi szakaszok röviden felvetnek néhány kulcsfontosságú szempontot, és egy-egy példa REST szolgáltatási szerződés tervezetet nyújtanak minden szolgáltatási modellhez.

Feladatszolgáltatások

Feladatszolgáltatások általában kevés szolgáltatási képességgel rendelkeznek, néha csak egyetleneggyel. A szolgáltatások célja automatizált üzleti folyamatok végrehajtása. További képességeket adhatunk hozzá az aszinkron interakciók támogatásához. REST-alapú feladatszolgáltatások gyakran POST kéréssel aktiválódnak, de ez nem mindig megbízható. Ahhoz, hogy bemenetet adjunk egy paraméterezett feladatszolgáltatáshoz, ésszerű az azonosítókat beilleszteni a képesség forráz azonosító sablonába. Ha a feladatszolgáltatás automatizál egy hosszú futási üzleti folyamatot, köztes választ ad a fogyasztónak. Ha további képességeket tartalmaz a vállalkozási folyamat állapotának ellenőrzésére vagy interakciójára, általában hivatkozásokat tartalmaz az elsődleges válaszkérelemmel kapcsolatos erőforrásokhoz.

Entitásszolgáltatások

az üzleti entitásokkal kapcsolatos funkcionális határokat állítanak fel. A szolgáltatások főként az entitáshoz kapcsolódó adatok feldolgozására összpontosítanak, általában idempotens és megbízható módszereket használnak.

Segédprogram szolgáltatások

általában agnosztikusak és újrafelhasználhatóak, de funkcionális határukat sokszor nem előre meghatározott.

Erőforrás azonosítók tervezése és szabványosítása

Erőforrás-azonosítók tervezése és szabványosítása kulcsfontosságú az hatékony REST szolgáltatási szerződés tervezésében. A szerkezet és a szókincs szabványosítása segíti az embereket az erőforrások és képességeik értelmezésében. Rugalmas erőforrás-azonosítók csökkentik a negatív összekapcsolódást és támogatják a vissza- és előrefelé kompatibilitást. A szolgáltatások fogyasztóinak lehetőséget kell biztosítani az információk beillesztésére az URL-ekbe, ami kritikus a szolgáltatások közötti összekapcsolódás szempontjából.

Szolgáltatásnevek az Erőforrás-azonosítókban

Szolgáltatásnevek az Erőforrás-azonosítókban: Az erőforrásokat szolgáltatásnevekkel azonosítani lehet. Az erőforrás-azonosítók struktúrája és szókincse szabványosításának célja többek között a könnyebb értelmezés és a negatív összekapcsolódás csökkentése. A szolgáltatásnév és az irányítás szinonimának kell lennie a maximális szolgáltatás autonómiához.

Más URI összetevők

Az URI út- és lekérdezési részei kontextust adnak egy adott szolgáltatás képességeihez. A {fragment} összetevő sosem kerül elküldésre a szolgáltatásnak, csak útmutatást tartalmaz a válasz feldolgozásához. Ha hiányoznak az URI összetevői, az URI relativvé válhat, és az URI kontextusa határozza meg, mit mutat. Relative URI-k gyakran hasznosak a kapcsolódó erőforrásokhoz való hivatkozáshoz anélkül, hogy további kontextusra lenne szükség.

Erőforrás-azonosító Átfedések

Az erőforrások lehetnek bármi, ami kapcsolódik a szolgáltatáshoz, például entitások, feladatok, sorok, stb. Az erőforrások azonosítói általában annyi kontextust tartalmaznak, amennyi szükséges a fogalmuk megjelöléséhez. Például egy erőforrás lehet "a mai vancouveri időjárás". Az erőforrások fogalmai néha átfednek, így ugyanaz az adat más erőforrás-azonosítókon keresztül is elérhető lehet. Az erőforrások kontextusa lehet dinamikus vagy munkamenet-specifikus. Például egy banki számla tranzakcióit azonosító URI lehet a fogyasztó számára az eddig kibékített és a kibékítetlen tranzakciók közötti helykitöltő. Az erőforrásokba beépíthetők lekérdezések is, például hőmérsékleti tartományok vagy a maximális hőmérséklet értéke egy adott dátum után.

Erőforrás-azonosító tervezési útmutatók

Néhány tipp az erőforrás-azonosítók SOA-támogatásában való optimalizálásához:

• Kerüljük az URL változó részének az első útszegmensbe, vagy statikus útszegmens előtt való helyezését, mert ez nehezíti a névtér kiterjesztését.

• A végződő perjelek gyakran gyűjteményt jelölnek. Egy GET kérés egy végződő perjellel záruló URL-hez az erőforrások listáját, míg egy POST új erőforrást hoz létre.

• Egyszerűsítsük a kanonikus neveket az erőforrás-azonosítókban, kerüljük a speciális karaktereket.

• Mindig hivatkozzunk a kanonikus nevekre teljes erőforrás-azonosítóval.

• Különítsük el az emberi felhasználók vagy szolgáltatás-fogyasztók beillesztett lekérdezési paramétereit az URL lekérdezési részében.

• A változók beillesztése az URL-mintákba kívánatos kapcsolódást eredményezhet, ezért minden változónak megegyezett jelentést kell kapnia a szolgáltatás és a fogyasztók között.

A REST korlátokkal való tervezés és szabványosítás

A REST korlátok különálló és elkülönült tervezési szabályokat jelentenek megfelelő tervezési célokkal. Fontos azonban tisztázni, hogy minden korlátot szigorúan alkalmazni kell-e. Mivel kiemelten fontos a szolgáltatások egységes kialakítása, javasolt a REST korlátok alkalmazásának egységes szabványosítása.

Állapotmentesség

Az Állapotmentesség korlátának két alapvető értelmezése van:

Az egyik lazább értelmezés szerint az ülési állapot olyan adat, amelyet egy kérésüzenet hivatkozhat, de nincs explicit erőforrás-azonosítója. Ennek az értelmezésnek megfelelően az ülési állapotot erőforrás-azonosítóval lehet ellátni a szolgáltatásban, hogy átalakíthassa szolgáltatási állapottá. Ezt követően az adatot adatbázisba vagy más tárolóba lehet elhelyezni. A további kérések (ugyanattól a fogyasztótól vagy más fogyasztóktól) az állapotát az erőforrás-azonosítójával hivatkozzák, így függetlenül megérthetők a korábbi kérésektől.

A szigorúbb értelmezés szerint az ülési állapot olyan adat, amely egy adott szolgáltatásfogyasztóhoz kötődik, és általában meg kell semmisíteni, amikor az ügyfél befejezi a folyamatban lévő szolgáltatási tevékenységet vagy interakcióját. Ennek az értelmezésnek megfelelően az adat erőforrás-azonosítóval való társítása nem alakítja át szolgáltatási állapottá, és nem szabad a szolgáltatás által tárolni a kérések között.

Gyorsítótár

Ez a korlát azt jelenti, hogy a válaszokat, amelyeket lehetséges újrahasználni a későbbi kérésekhez, gyorsítótárvezérlési metaadatokkal kell ellátni. Ez a korlát főként az adatlekérdezési módszerekre vonatkozik, például a GET és HEAD. Azonban néhány POST és más típusú kérések is alkalmazhatók, amelyek elsősorban az adatok lekérdezésére irányulnak. A gyorsítótár két alapvető formája létezik: egy válaszüzenet újrahasználható egy bizonyos időszakra vagy csak akkor újrahasználható, ha annak érvényességét minden használatkor ellenőrizni kell. Ahhoz, hogy eldönthesse, próbálja-e újrahasználni a gyorsítótárazott választ, a gyorsítótárnak mechanizmusra van szüksége annak meghatározásához, hogy két kérés egyenértékű-e a gyorsítótárazás céljából. A HTTP Vary fejléc használata segíthet az egyes kérések ekvivalenciájának azonosításában.

Egységes Szerződés

Az HTTP módszereket és média típusokat "szabványnak" kell tekinteni, ami a "gyakorlatban való újrahasznosítást" jelenti több szolgáltatás részéről. Az egységes szerződés azt jelenti, hogy az egyes szolgáltatásoknak közös módszereket, média típusokat és egyéb üzenetelemeket kell használniuk a szolgáltatási szerződésekben. A tervezési normáknak az új módszerek és média típusok azonosítását és követését kell biztosítaniuk.

Kulcsfontosságú pontok összefoglalása

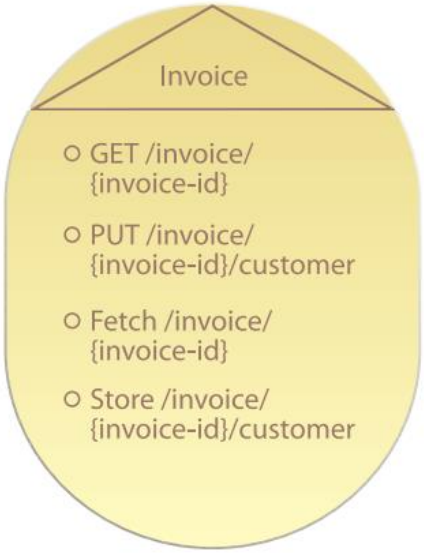
• Az egységes szerződés módszerek és média típusok újrahasznosításának meghatározása a szolgáltatási állomány felelőssége, ahogy ennek az egységes szerződés elemeknek a REST korlátoknak való megfelelőségének betartatása is a tervezési normák részeként.

• A különböző szolgáltatásmodelleken alapuló szolgáltatások általában más-más szolgáltatási szerződés-tervezési szempontokat és jellemzőket fognak bevezetni.

• Az erőforrás-azonosítók használata szabványosítható egy adott szolgáltatási állományban mind a szintaxis, mind a szókincs szintjén.

10.3 komplex módszertervezés

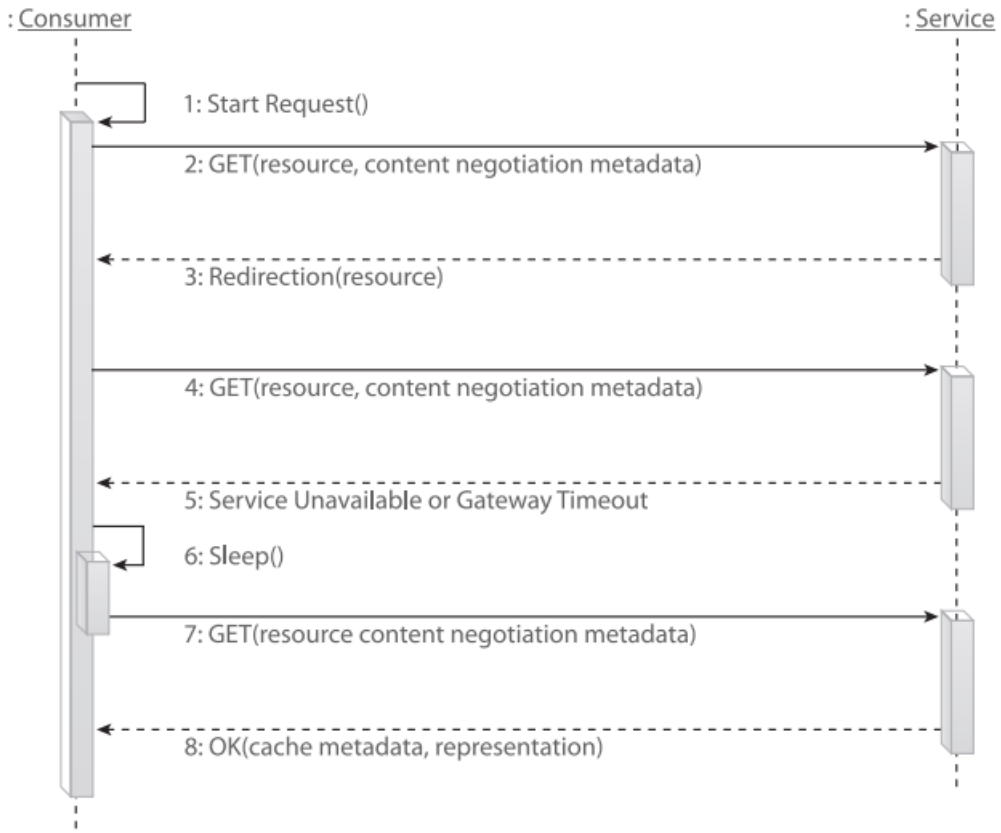
A uniform szerződés alapvető módszereket állapít meg az adatkommunikációhoz, újra használható és szabványosítja a szolgáltatáskészletet. Az HTTP szabványosítása a weben csak alapvető működést biztosít, de egy saját IT-környezetben további testre szabásokra van lehetőség. Például a számviteli dokumentumok lekérdezését külön szabályok szerint lehet végrehajtani, amelyek az alap HTTP módszereket kiegészítik.



A komplex módszer előre meghatározott interakciókat foglal magában egy szolgáltatás és egy fogyasztó között. Ezek a módszerek több alapvető HTTP módszer összetételéből vagy azok többszöri alkalmazásából állnak, és további funkcionalitást vezethetnek be. A komplex módszereket általában egy adott szolgáltatáskészleten belül testre szabják és dokumentálják, ami lehetővé teszi az összes szolgáltatás számára való implementációjukat. A szolgáltatások a komplex módszerek támogatását külön képességként jelzik, ami segíti a fogyasztóprogramok fejlesztését.

A komplex módszerek használata nem kötelező, és csak akkor javasolt, ha kontrollált környezetben biztonságosan definiálhatók és alkalmazhatók a növelt interoperabilitás céljára. Döntésünket az üzleti követelmények és a szolgáltatáskészlet sajátosságai alapján kell meghozni. A komplex módszereknek nevével ellentétben egyszerűséget hoznak a rendszerbe, de a használatukat átgondoltan kell megtervezni.

Amikor egy szolgáltatás irányítási kódot küld vissza, nem biztos, hogy minden fogyasztó megfelelően reagál rá, és egy ideiglenes kommunikációs hiba váratlan következményekkel járhat. Hiányzó irányelvek felesleges redundáns üzenetfeldolgozási logikához vezethetnek. A komplex módszerek használata bonyolult architektúrát eredményez, amit el kell kerülni. A következő szakaszokban bemutatásra kerül egy sor példa komplex módszer, amelyek két kategóriára oszthatók: Állapotfüggetlen és Állapotfüggő. Fontos megjegyezni, hogy ezek a módszerek nem iparági szabványok, hanem általános funkcionalitások kezelésére lettek testre szabva.



Állapotfüggetlen komplex módszerek:

Ezek olyan módszerek, amelyek az állapotfüggetlenség elvét követik. A "Lekérdezési módszer" például lehetőséget ad automatikus újra próbálkozásra időtúllépés vagy kapcsolódási hiba esetén, valamint támogatja a futási idejű tartalom-tárgyalást és az átirányítást a szolgáltatásszerződés változásainak kezelésére.

Raktározási módszer:

Amikor új erőforrásokat adunk hozzá, meglévők állapotát módosítjuk vagy régieket törlünk, az alap PUT vagy DELETE módszerekkel, a szolgáltatásfogyasztók időtúllépést vagy kivételt kaphatnak. Ezt a viselkedést szabványosítani és optimalizálni lehet egyedi Raktározási módszerrel, amely automatikus újra próbálkozást, tartalom-tárgyalást és átirányítási támogatást biztosít. Fontos megjegyezni, hogy ezek a kérések a szolgáltatás logikájára vonatkoznak, nem pedig az adatbázis műveleteire.

Delta módszer:

Ez egy szinkronizációs mechanizmus, amely a változó erőforrások állapotának követésére szolgál a szolgáltatás és a fogyasztók között. A szolgáltatás nyomon követi az erőforrások változásait, a

fogyasztó pedig a történeti változásokat kérdezheti le. A delta erőforrás lehet változások listája, jelezheti, hogy nincs változás, vagy hogy a delta lejárt. Fontos, hogy ezek a kérések a szolgáltatás logikájára vonatkoznak, nem pedig az adatbázis műveleteire.

Async módszer:

Ez egy előre meghatározott, összetett módszer, amely lehetővé teszi az aszinkron üzenetek sikeres és megszakított cseréjét. Segít abban, hogy olyan kéréseket kezeljünk, amelyek hosszabb időt vesznek igénybe, mint amit a standard HTTP időtúllépések engednek. A szolgáltatás egy visszahívási mechanizmust biztosít a fogyasztó számára, amely lehetővé teszi a kérések állapotának követését és az eredmények visszakapását. Ha a kérés túl sokáig tart, a szolgáltatás megszakítja a feldolgozást, és nincs válasz. A fogyasztók a kérés állapotát és eredményét később is lekérdezhetik.

Állapotfüggő komplex módszerek:

Ezek olyan összetett módszerek, amelyek a REST alapján tervezett szolgáltatásokba építenek olyan interakciókat, amelyek szándékosan áthágják az Állapotfüggetlenség elvét. Habár ezek a módszerek gyakoriak a hagyományos vállalati alkalmazásokban, a kommunikációjuk nem jellemző a Világhálón. Állapotfüggő módszerek használata esetén elfogadjuk a skálázhatóság csökkenését.

Trans módszer:

Ez a módszer két fázisú kötelezettségvállalást tesz lehetővé egy szolgáltatásfogyasztó és egy vagy több szolgáltatás között. A tranzakció keretein belül végrehajtott változások vagy sikeresen átterjednek az összes részt vevő szolgáltatáson, vagy minden szolgáltatást visszavonnak az eredeti állapotába. A módszer egyedi funkciókat igényel a résztvevőktől, és általában egy tranzakció-vezérlőt is magában foglal. Az állapotfüggetlen tranzakciós modelleket részletesebben tárgyalják a 12. fejezetben.

PubSub módszer:

Ez egy olyan komplex módszer, amely lehetővé teszi a kiadás-feliratkozás alapú kommunikációt, ami szándékosan megsérti az Állapotfüggetlenség elvét. Segít valós idejű interakciók támogatásában, ahol az eseményekre azonnal reagálni kell. A mechanizmus könnyűsúlyúnak tekinthető, mert nem szükséges a változások valós idejű kiküldése, csupán az erőforrások megváltozásáról történő értesítés, majd az új adatok letöltése a szolgáltatásfogyasztók részére.

Fontos pontok összefoglalása:

• Mind a uniformis szerződés, mind az egyedi szolgáltatási szerződések tervezésekor fontolóra vehetjük a komplex módszerek létrehozását a szerződések által kínált funkciók részeként.

• A komplex módszerek magukban foglalják több egyszerű HTTP módszer összeszervezését vagy egyetlen egyszerű HTTP módszer ismételt végrehajtását, valamint más funkcionális jellemzőket, amelyek a meghatározott üzenetinterakciók részét képezik.

• Az ideális esetben a komplex módszereket szabványosítjuk, hogy az interakciós viselkedés minden szolgáltatásban és fogyasztóban, amelyek használják őket, következetes legyen.

• Mind a "stateless" mind a "stateful" komplex módszerek tervezhetők, bár az utóbbi változat nem teljesen megfelelő a REST-nek.

II.) A webet támogató ipari szabványok

(387-390 Industry Standards Supporting the Web)

Egy REST-stílusú architektúrában az alkalmazásprotokollt az egységes szerződés határozza meg. A Web alkalmazási rétegének három alkotó szabványa az Egységes Erőforrás-azonosító (URI), a Hiperszövegátviteli Protokoll (HTTP) és az Internet Média típusok (MIME).

Ezeket a részleteket külön-külön határozzák meg, lehetővé téve a specifikációk független kiadási ütemezését, más specifikációkkal való kombinálását a Web részeként, vagy más kontextusokban való felhasználását.

Az HTTP alkotja a Web alkalmazási rétegét, és általában a Transmission Control Protocol (TCP) és gyakran a Transport Layer Security (TLS) protokollra, más néven Biztonságos Szálas Csomópont (SSL) épül. Ezek a protokollok pedig különböző verziókon keresztül épülnek az Internet Protokoll (IP) és többféle Linkrétegű protokollra.

Az előre meghatározott Internet média típusok halmaza olyan sémákat tartalmaz, amelyek segítségével az adatokat olyan módon lehet kódolni, hogy azokat széles körben értelmezhetők legyenek a különböző címzettek számára. Néhány média típus az Kiterjeszthető Jelölőnyelv (XML) alapján működik, mások az idősebb Szabványos Általános Jelölőnyelv (SGML), a JavaScript Objektumok Jegyzéke (JSON) alapján, vagy különböző bináris alapokon.

Az URI specifikáció támogatja mind az alkalmazásprotokollt, mind a média típusok specifikációit. Ez a szintaxis lehetővé teszi az erőforrások azonosítását egy generikus tartály segítségével, amelyet közvetlenül üzenetcserére használhatunk azokkal a szolgáltatásokkal, amelyek mindegyik erőforráshoz tartoznak.

Az Internet Engineering Task Force (IETF)

Az IETF a fő testület az interneten, felelős az alkalmazásprotokollok, szállítás, biztonság és más területekért. Az IETF szabványosítási folyamata hatalmas, de jól szervezett. Az IANA csoport szabályozza az internetes azonosítókat és portszámokat. Az IETF által szabályozott kulcsfontosságú szabványok közé tartozik az HTTP és a TCP. Az IANA regisztrálja a média típusokat.

Az IETF által szabályozott kulcsfontosságú szabványok közé tartozik:

• Hiperszövegátviteli Protokoll (HTTP) – http://www.ietf.org/rfc/rfc2616

• Egységes Erőforrás-azonosító (URI) – http://www.ietf.org/rfc/rfc3986

• Atom Szindikációs Formátum – http://www.ietf.org/rfc/rfc4287

• Transmission Control Protocol (TCP) – http://www.ietf.org/rfc/rfc793

További média típusokat az IANA regisztrál: http://www.iana.org/assignments/media-types/

The World Wide Web Consortium

A World Wide Web Consortium (W3C) eredetileg az IETF-nek szánt specifikációkat dolgozott ki, de később saját szabványokat is hozott létre, például az HTML, XML, XML Schema és mások területén. A tagok széleskörűen részt vehetnek különböző csoportokban, munkacsoportokban és eseményeken. Az Advisory Board irányítja a stratégiai és menedzsment kérdéseket, míg a Technical Architecture Group (TAG) az internetes architektúra elveivel foglalkozik. A W3C munkája

nyilvánosan is elérhető, így a közönség részt vehet a vitaáramlatokban és munkacsoportokban. A JSON és a JavaScript szintén fontos ipari szabványok.

Más Webes Szabványok:

Az alábbi iparági szabványok is érdemesek megjegyzésre:

JavaScript Object Notation (JSON)

JavaScript, ECMA International.