Фреймуърк системи за уеб програмиране

Лекция 4:

REST

Съдържание

- Въведение
  - Уеб и REST
- Технологии и имплементация
- REST принципи
- REST имплементация
- Заключение

Въведение

Representational State Transfer (REST) е архитектурен стил за изграждане на разпределени системи. Уеб е пример за такава система. REST-стилови приложения могат да бъдат построени посредством много различни технологии. Основните принципи на REST са тези на ресурсно-ориентираните състояния и функционалности, идеята за еднозначен начин за идентифициране на ресурсите, и идеята как операциите над тези ресурси са дефинирани от гледна точка на един единствен протокол за взаимодействие с ресурсите. REST-ориентираният системен дизайн може да доведе до системи, които са отворени, скалируеми, разширяеми, и лесни за разбиране.

Уеб и REST

- Уеб е една разпределена система за хипер-медиа
  - Основните архитектурни компоненти са URI, HTTP, и HTML
  - Всички други уеб технологии са построени на тази основа
  - Ако не са, има голяма вероятност да не са добре-построени уеб технологии
- Уеб е удивително отворен, скалируем, разширяем, и лесен за разбиране
- *Отвореността* позволява нови технологии да бъдат създавани
  - *скалируемостта* се грижи за това системата да няма слаби места (bottlenecks)
  - *разширяемостта* позволява на Уеб да еволюира без да се прави на ново дизайна на съществуващите части
  - *простотата* подсигурява, че системата ще оцелее и ще се развива
- Никоя друга информационна система не може да се доближи до Уеб
  - Но не всички дизайни на информационни системи могат да приемат ограниченията на Уеб
  - REST би трябвало да се възприема като примерно ръководство как да построяват истински Уеб приложения
  - Други приложения ще продължат да се построяват посредством други методи

Софтуерен дизайн

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.

C. A. R. Hoare, The Emperor's Old Clothes, 1980 Turing Award Lecture

Има два подхода при софтуерният дизайн: Единият е да се построи толкова прост дизайн, че очевидно да няма недостатъци; Другият е да се построи толкова сложен дизайн, че да няма очевидни недостатъци. Първият метод е далеч по-труден.

Обектно ориентиране

- Обектната ориентация е стил в *софтуерното инженерство*
  - Може да бъде приложен към всеки програмен език
  - В зависимост от езика, това е повече или по-малко лесно
  - ОО езиците поддържат или дори налагат определени шаблони за дизайн
  - *Спагети код* може да бъде писан на всеки програмен език
- Имплементациите могат да бъдат лоши или добри
  - Качеството на имплементацията зависи от програмиста
  - Програмистите могат да бъдат по-подкрепяни и контролирани в език ОО език
  - Метриките за качеството на имплементацията трябва да са базирани на продукта, не на езика на който е писан
- Добри програмисти винаги пишат добър код?
- Лоши програмисти винаги пишат лош код?
- Всичко програмисти имат нужда от добри инструменти за да създават високо-качествен код

Технологии и инструменти

- Технологиите помагат за решаването на задачи
  - Построени са с определени цели в предвид
  - Специализирани са в решаването на задачи по определен начин
- Технологичните избори са много важни
  - Технологията (т.е. инструмента) оформя как дадена задача се решава
  - Да се работи по начин в противоречие с диктуваният от инструмента е възможно, но трудно и рядко се прави
- Технологиите понякога замъгляват по-важните неща
  - Задачата трябва да е добре-дефинирана и напълно осъзната
  - Основният подход за решаването на задачата трябва да е идентифициран
  - Технология, която поддържа този подход, трябва да бъде избрана

Имплементациите са продукти

- Имплементациите са построени на (и оформени от) технологиите
- Имплементация = Концепции + Технологии
- Качеството на имплементация зависи и от двата фактора
  - Правилният избор на технология е много важен
  - Отговорното използване на тази основа е също толкова важно
- Добрият REST е труден за възприемане, както е и доброто ОО
  - Пдорукти могат да трърдят че са REST/ОО
  - Може да използват технологии, които поддържат REST/ОО
  - Само вминателно изследване може да разкрие истинността на това твърдение
  - В повечето случаи това се случва само, когато продукта трябва да се промени

Дефиниция

- Ресурсите са дефинирани от URI-та
- Ресурсите са манипулирани посредством техните репрезентации
- Съобщенията са само-описващи се и не запазват състояние
- Може да има много репрезентации на един ресурс
- Състоянието на приложението се управлява чрез ресурсна манипулация

Ресурси

- Ресурсите са дефинирани от URI-та
  - Ресурсите не могат да бъдат достъпени или манипулирани директно
  - REST работи с ресурсни репрезентации
- Ресурсите са всички неща с които искаме да работим
  - Ако не може да се именува нещо, не може да се прави нищо с него
  - Популярен тип ресурси в Уеб са документите
  - Документите най-често са структурирана колекция от данни (информация)
- Документите са абстрактна концепция за описателни ресурси
  - Могат да бъдат използвани в различни контексти (напр. формат)
  - Различни приложения може да се интересуват от различни репрезентации
  - В основите самият ресурс си е един и същ (с различни репрезентации)

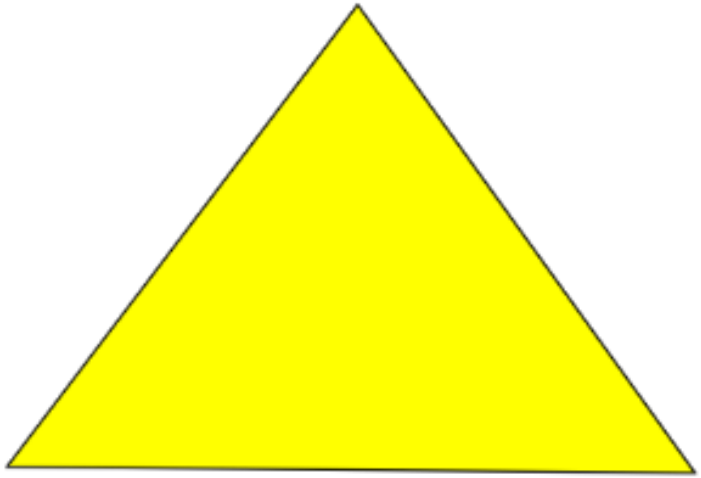
Състояние

- Състоянието се описва като част от съдържанието, което се обменя
  - Прекъсвания в работата на сървъра не създават проблеми за клиента
  - Възможно е да се превключва между различни сървъри за различни действия
  - Клиентите могат да съхранят репрезентацията, за да запишат състоянието
- Прехвърлянето на състоянието прави системата скалируема
  - Преноса на данни не е обвързан със състояние (няма нужда от запазване на състоянието при обработка на съобщенията)
  - Състоянието се прехвърля между клиента и сървъра

Установяване на общ модел

- Разпределените системи трябва да са базирани на споделен модел
  - Традиционните системи трябва да се съгласуват за общ съвместен API
  - REST системите структурират договорността в три области
- REST е построен около идеята за опростяване на договорността
  - *съществителни* се използват за да се именуват ресурсите, за които може да се говори
  - *глаголи* са операциите, които могат да се приложат към именуваните ресурси
  - *Типовете съдържание* дефинират кои информационни репрезентации са достъпни

REST триъгълник



Nouns
(Unconstrained)
eg http://wikipedia.org/

Verbs
(Constrained)
eg GET

Content Types
(Constrained)
eg HTML

Съществителни

- Съществителните са имената на ресурсите
  - При повечето дизайни, тези имена са URI-та
  - URI дизайна е много важна част от REST-базираният системен дизайн
- Всичко значимо би трябвало да е именувано
  - Поддържайки добре-създадени имена, приложенията могат да говорят за именувани неща
  - Нови операции и репрезентации могат да бъдат създавани
- Отделянето на съществителните от глаголите и репрезентациите подобрява разширяемостта
  - Приложенията могат да продължат да работят с ресурси, дори и да не могат да ги обработват на 100%
  - Въвеждането на нови операции в Уеб не го чупи
  - Въвеждането на нови типове съдържание в Уеб не го чупи

Глаголи

- Операции които могат да бъдат прилагани върху ресурси
- Основната идея на REST е да използва само *универсални глаголи*
  - Универсалните глаголи могат да бъдат приложени върху всички съществителни
- За повечето приложения, основните методи на HTTP са достатъчни
  - GET: Прочитане на ресурс (трябва да няма странични ефекти)
  - PUT: Прехвърля ресурс към сървъра (презаписва вече съществуващ такъв ако има)
  - POST: Добавя към съществуващ ресурс на сървъра
  - DELETE: Унижтожава ресурс (името му не може да се използва повече)
- Отговарят на най-популярните основни операции с бази данни
  - CRUD: Create, Read, Update, Delete (Създаване, Четене, Промяна, Изтриване)

POST-ване

- POST добавяне, вместо са презаписващо обновяване
- POST може да има различни ефекти
  - Чрез POST-ване, състоянието на съществуващ ресурс се променя и нов ресурс се създава
  - Чрез POST-ване, само съществуващият ресурс се променя
  - Сървъра сигнализира разликата посредством HTTP отговори (200 OK или 201 Created)
- Това е *дизайн избор*
  - Ако добавената информация трябва да може да се достъпва независимо, се създава нов ресурс
  - За промени изцяло в съществуващи ресурси, няма нужда от създаването на нови
- Всички ресурсите трябва да са достъпни чрез URI
  - Ако е уместно, дадена релация може да се репрезентира във формата на ресурса(ите)

Типове съдържание (content types)

- Репрезентациите би трябвало да могат да се обработват от машина
  - Не е задължително, може да са непрозрачни за приложенията
  - В много случаи, репрезентации които могат да се обработват от машини са за предпочитане
- Ресурсите са абстракции, REST подава и приема само репрезентаци
  - Ресурсите могат да имат различни репрезентации (типове съдържание)
  - Клиентите могат да заявят от кои типове съдържание се интересуват
- Добавянето или променянето на типовете съдържание не променя архитектурата на системата
  - Различни клиенти и сървъри могат да поддържат различни типове съдържание
  - Договаряне на съдържанието (Content Negotiation) позволява типовете съдържаниет да бъдат договаряни динамично

REST и Уеб услуги

- REST е описание на основните дизайн принципи на Уеб
  - Не е нещо ново, то е систематичен изглед на Уеб
  - REST постулира, че може да се учи от успеха на Уеб
- Уеб услугите (в тесният смисъл) не строят над REST
  - Те използват HTTP като транспортен протокол
  - Те пресъздават Уеб функционално посредством допълнителни спецификации (WS-*)
  - Те са построени от програмисти използвайки подхов в посока отгоре надолу
- REST и Уеб услугите имат различни подходи при дизайна
  - REST започва от ресурсите и поема всичко от там
  - Уеб услугите се фокусират на съобщенията, които в повечето случаи са операции / действия

REST технологии

- REST не е обвързан с конкретно множество от технологии
  - URI-та са най-разпространеният избор за съществителни
  - HTTP методите са най-разпространеният избор за глаголи
  - XML и JSON са най-разпространените избори за съдържание
- Избирането на други технологии би трябвало да е поради много добра причина
  - Построяването на REST система би трябвало да я направи отворена и достъпна
  - Технологичните избори са също толкова вачни, колкото и архитектурните

URI

- REST изисква много URI дизайн
  - Вместо да бъдат създавани като страничен ефект, те са в основата на системата
- Дизайнът на URI-та и започването от тях е нов начин на мислене
  - URI-тата са много повече от просто адресът на Уеб страница
- URI-тата са имена на концепции
  - Концепциите не се предават, само техните репрезентации
  - Фокусирането на концепции вместо на репрезентации е полезно

HTTP

- HTTP е най-успешният RESTful протокол
  - Roy Fielding (един от основоположниците на HTTP), полага основите на термина REST в дисертацията си
- HTTP би трябвало да се използва като протокол на приложно ниво
  - Уеб услугите използват HTTP като транспортен протокол
  - HTTP има много повече да предложи от пробиваща защитните стени тръба
- Уеб инфраструктурата е построена чрез правилното използване на HTTP
  - Кеширането е вградено в HTTP и кешовете оптимизират Уеб транспарентно
  - Идентификацията (authentication) може да се извърши посредством средствата за идентификация на HTTP
  - Защитен пренос на данни може да се осъществи посредством използването на HTTP върху SSL (HTTPS)

XML

- URI-идентифицируемите ресурси са абстрактни концепции
  - За обработка от машини, XML е добра репрезентация
  - За човеко-ориентирани взаимодействия, HTML вероятно е по-добрият избор
- Свързването с други ресурси трябва да се прави посредством URI
  - XML няма вградени предположения за идентификаторите
  - Но поддържа URI, например с *XLink*, *XInclude* и *XML Base*
  - RESTful приложенията са за навигирането на мрежа (Уеб) от URI-идентифицирани ресурси

Заключение

- REST е архитектурен стил
  - URI/HTTP/HTML/XML могат да бъдат заменени
  - Основният принцип за ресурсно-базирано взаимодействие остава валиден
- RESTful системният дизайн може да създаде по-добри системи
  - Изисква малко повече усилия в първоначалният дизайн
  - Предпоставя за по-малко главоболия в по-късните етапи
- SOA често не са наистина RESTful
  - SOA се фокусира на операции
  - REST се фокусира на ресурси
- RESTful дизайна е добра отправна точка за ОО имплементации