Script **generated by TTT**

Title: Distributed_Applications (03.07.2012)

Date: Tue Jul 03 14:34:57 CEST 2012

Duration: 85:50 min

Pages: 28

Web Services

Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the W3C community.

Motivation - Example

Service Oriented Architecture - SOA

Web Services - Characteristics

Web Services Architecture

Simple Object Access Protocol (SOAP)

Web Services Description Language (WSDL)

Universal Description, Discovery, and Integration (UDDI)

REST

Web Service Composition

Adopting Web Services

Mashups

Generated by Targeteam

REST

REST (Representational State Transfer) is an architectural style of distributed applications.

- REST is not a standard; it is a set of principles how to use Web standards, such as HTTP, URIs and Mime Types.
- The Web is a REST system.

REST is based on the following key principles

- give every relevant resource an ID: use URIs to identify everything that is any item of interest.
  - URL: http://www.boeing.com/aircraft/747
- A representation of the resource is returned (e.g., Boeing747.html). The representation places the client application in a state.
- link resources together: navigating links results in state transfers of the client application.
- use standard methods: such as get, post, put, delete.
- communication is stateless.
- resources with multiple representations: client may specify the formats which it accepts
  - GET /customers/1234 HTTP/1.1
  - Accept: text/x-vcard

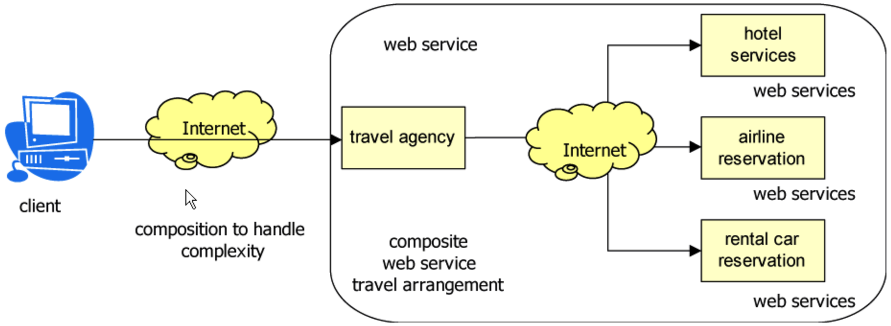
Generated by Targeteam

Web Service Composition

an important issue is the choice of the appropriate granularity

- small vs. large Web Services - thousands vs. a handful of Web Services
- what are the appropriate reusable, shared business components

Composition of complex Web Services from smaller reusable Web Services



Dimensions to handle complexity

Web Service Orchestration

Generated by Targeteam

component model: defines the sub-services.

orchestration model: defines the order in which the sub-services are invoked.

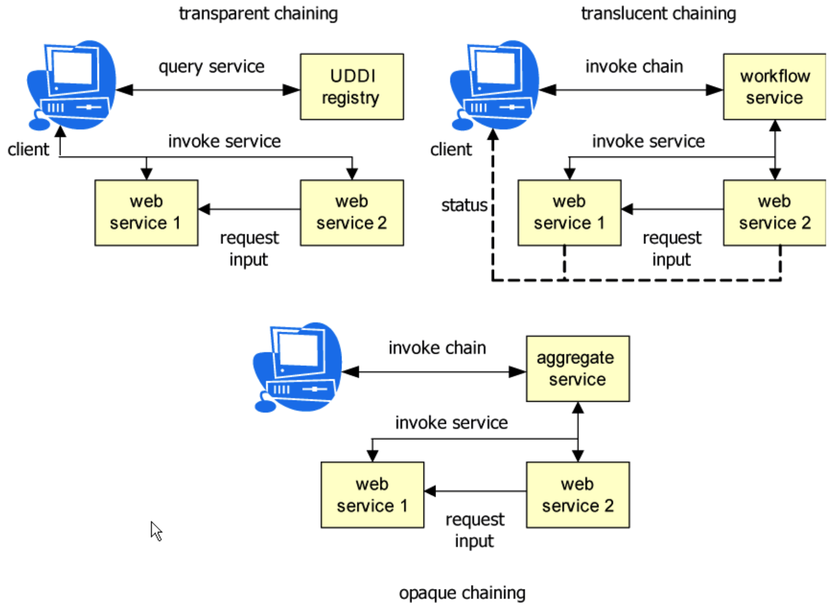
- WS-Coordination is an extensible framework that describes how different Web Services work together reliably. Coordination framework contains
  - Activation, Registration and Coordination services

data access model: specifies the data exchange between the sub-services.

transactional model: transactional semantics of the composed service.

- WS-Transaction specifies the protocols for each coordination type (used by WS-Coordination)
  - AtomicTransactions: all-or-nothing property, 2-phase-commit.
  - Business Activity: handle long-lived activities and to apply business logic to handle business exceptions; BusinessAgreement Protocol.

exception handling: handling of errors in the sub-services.



transparent chaining

Web Services

Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the W3C community.

Motivation - Example

Service Oriented Architecture - SOA

Web Services - Characteristics

Web Services Architecture

Simple Object Access Protocol (SOAP)

Web Services Description Language (WSDL)

Universal Description, Discovery, and Integration (UDDI)

REST

Web Service Composition

Adopting Web Services

Mashups

Example Web Services

The following lists some available Web services; often registration necessary in order to use them.

Amazon E-Commerce Service (ECS)

FedEx Office and Priniting Service

- printing of online documents and distribution of paper documents as commercial Web Service
- free print plug-in for standard office application; plug-in added to list of printers
- Pick up your document at any U.S. location or ship via FedEx for added convenience.
- location independent use of printing service

Via Michelin

- Reverse Geocoding Web Service allows users to obtain the closest road segment (named or not) for each supplied geographic coordinates (WGS84).

XMethods : clearinghouse for Web Services

ECS provides access to Amazon's product database with the following types of data

- detailed product information.
- customer-contributed content, e.g. wish list, product reviews.
- seller information.
- 3rd party product information.
- shopping cart contents.

ECS supports both SOAP and REST style interactions.

product operations

- ItemSearch: performs a search for a specific item, typically using a set of keywords
- SimilarityLookup: returns a list of similar products to a given product-ID (based product specifications and features).
- ItemLookup: access to the data related to a specific product.

remote shopping cart operations

- CartCreate: creates a remote shopping cart.
- CartAdd: adds an item to the shopping cart.
- further operations are CartGet (obtains content of the cart), CartModify (remove an item from cart)

The following lists some available Web services; often registration necessary in order to use them.

Amazon E-Commerce Service (ECS)

FedEx Office and Priniting Service

- printing of online documents and distribution of paper documents as commercial Web Service
- free print plug-in for standard office application; plug-in added to list of printers
- Pick up your document at any U.S. location or ship via FedEx for added convenience.
- location independent use of printing service

Via Michelin

- Reverse Geocoding Web Service allows users to obtain the closest road segment (named or not) for each supplied geographic coordinates (WGS84).

XMethods : clearinghouse for Web Services

Apache Axis supports an environment to implement and provide Web services.

- set of client-side APIs for dynamically invoking SOAP Web services (with or without WSDL descriptions).
- tools to translate WSDL documents into Java frameworks.
- mechanisms for hosting Web services either within a servlet container (e.g. Tomcat) or via standalone server.
- a set of APIs for manipulating SOAP envelopes, bodies, and headers, and using them inside Message objects.
- data binding which enables mapping of Java classes into XML schemas and vice versa.
- a transport framework that allows usage of a variety of underlying transport mechanisms (e.g. JMS, email, etc).

Axis2

In the meantime there exists already Apache Axis2

- Java-based implementation of both the client and server sides of the Web services
- Axis2 is more flexible, efficient and configurable in comparison to Axis1.x
- Axis2 not only supports SOAP messages, but it also supports RESTful Web services.

Java provides a number of APIs implementing the Web Services standards

SAAJ (SOAP with Attachments API for Java)

- SOAP messages as Java objects

JAX-WS (Java API for XML based Web-Services)

- programming model for Web Services; replaces JAX-RPC

JJWSDL: Accessing WSDL descriptions

JAXR (Java API for XML Registries)

- Accessing Web Services Registries, e.g. UDDI

JAXP (Java API for XML Processing)

- Abstract XML-API-Specification implemented by e.g. Apache Xalan(XSLT), Apache Xerces2 (XML Parsing (DOM, SAX..)).

XWSS (Java Web-Services Security)

- Signatures, Encryption (roughly for SOAP what SSL is for HTTP)

Adopting Web Services

There exist already a variety of free of commercial Web services; provided especially by Internet companies, such as Google, Amazon or Yahoo.

Integration and WS Standards

process management

process modeling, execution: BPEL4WS
process control : -
user interface integration : WSUI/WSXL

message management

transformation services : XSLT
synchronization : -
transaction services: WS-Coordination , WS Transactions

adapter

interface description: WSDL
messaging: SOAP

middleware

interface description : WSDL
reliability: WS Reliability
Messaging: SOAP, XML
Transport : HTTP, SMTP, ..

physical network

transport layer: TCP, UDP
network layer: IP

meta data & additional services

meta database : UDDI

system management: -

security services: WS-Security, SAML, XML-Encryption, XML-Signature

development support : -

Supporting - Restraining Forces

The adoption of Web Services in organization depends on

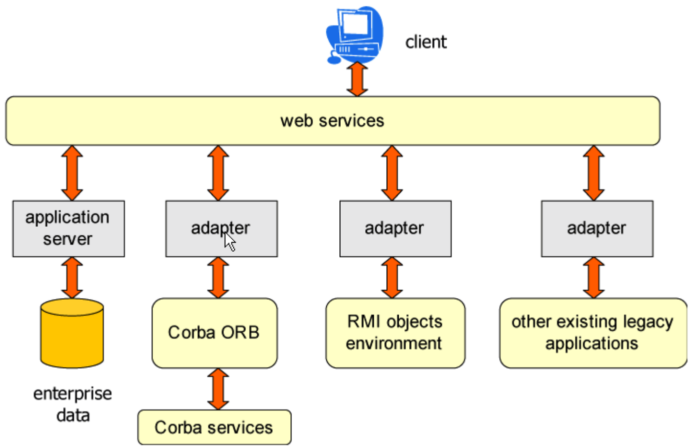
Supporting Forces

- interoperable networked applications, i.e. independence of hardware, operating system, application server, …
- easier exchange of distributed data
- easier access of enterprise wide data
- availability of external services, encapsulation of legacy applications
- cross-organizational computing
- reduced maintenance cost, easier reuse of components
- emerging industry-wide standard

Restraining Forces

- different formats and semantics of data sources
- security issues due to network access
- standards are evolving and not fixed yet
- lack of understanding of effects on operational systems

Distributed Process Architecture



Semantic Web Services

In order to allow for ***automatic discovery of appropriate*** web services and of ***automatic interaction / chaining / incorporation*** with web services

we need semantic meta-data for web services: Web-Service Ontologies, DataTypes with rich semantics....

Example: Map-Service

Input: (int, int)

Output: APPLICATION/GIF

Input: (int, int):

(x,y) of center of map ?

of corner of map ? which corner ?

what coordinate system? Wgs84? Gauss-Krueger? ...

Output: APPLICATION/GIF:

What kind of map? Topological? Political? POI? Traffic?

Units of measure?

candidate technology: OWL-S (Ontology Web Language for Web Services)

OWL-based Web service ontology, which supplies Web service providers with a core set of constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form.

Generated by Targeteam

Mashups

Definition: Mashup simply indicates a way to create new Web applications by combining existing Web resources utilizing data and Web APIs.

Mashup Techniques

Work for the combination of data and services can be done on the server, the client or both of them.

1. Mashing on the Web Server
2. Mashing using Ajax
3. Mashing with JSON

Development Support

Yahoo Pipes are hosted and executed on a Yahoo server.

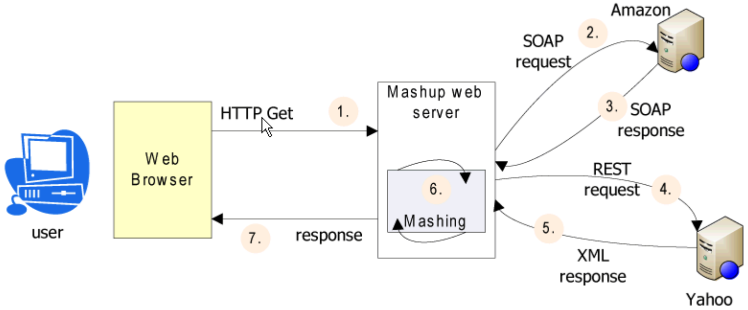
QedWiki was a Wiki-based mashup maker by IBM; pages are hosted on an IBM server; mostly executed on the client side.

ProgrammableWeb provides a mashup directory and marketplace which let users rank and discuss mashups.

Generated by Targeteam

1 Mashing on the Web Server

All the work of mashing is done on a Web server while the browser just waits for a response.

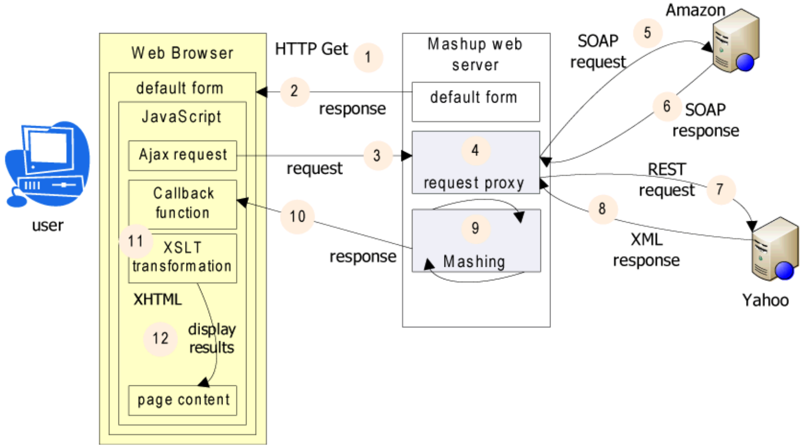


Characteristics

- Browser is decoupled from the partner sites supplying the data.
- Web server acts as a proxy and aggregator for the responses.
- Browser requests the entire page.
- Scalability problem because server does all the work.

Generated by Targeteam

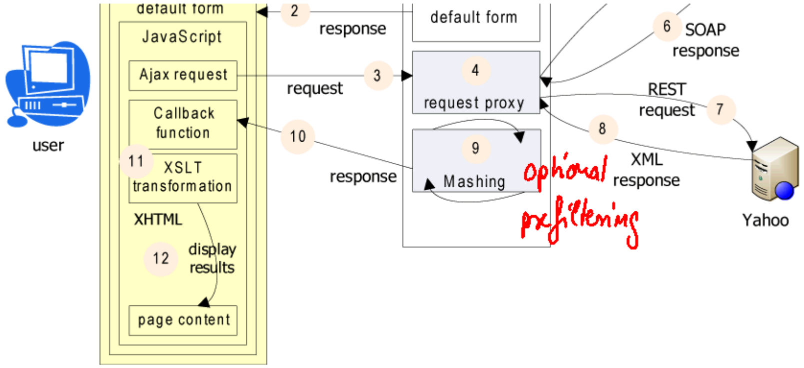
This approach allows a richer user experience; the work is divided between the server and the browser.



Characteristics

- more complex because developers face JavaScript challenges, server communication and asynchronicity.
- Ajax may refresh only a portion of the page.
- navigation mechanism of browser is bypassed.
- approach may result in a rich Internet application.

2 Mashing using Ajax

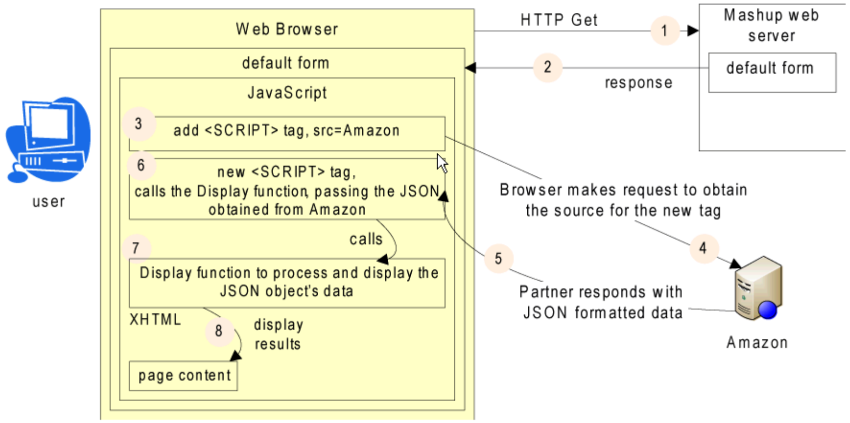


Characteristics

- more complex because developers face JavaScript challenges, server communication and asynchronicity.
- Ajax may refresh only a portion of the page.
- navigation mechanism of browser is bypassed.
- approach may result in a rich Internet application.
- presentation of results is driven by XSLT style sheet.
- browser is doing most of the work.
- all data are routed through a common point on the server.

Generated by Targeteam

JSON (JavaScript Object Notation): lightweight data-interchange format that is gaining popularity in the mashup community.



Characteristics

- the browser communicates directly with the partner site.
- programmers must handle pre-made objects supplied in JSON.
- JSON objects are easier to read than XML.
- there is no data consolidation on the server.

Development Support

in order to facilitate and speed up mashup development a number of tools and frameworks have recently emerged. Two dimensions may be distinguished

component model : describes the characteristic properties of the mashup components

a well-defined component interface facilitates reusability of components

component properties:

type: a component can be data, application logic or user interface.

interface: create-read-update-delete (CRUD) interface, API for a specific programming language or IDL/WSDL.

extensibility: whether the user may extend the component model.

composition model : specifies how the components are glued together to create the mashup application

flow-based: defines the orchestration as sequencing or partial order among components.

event-based: uses the publish-subscribe model.

Examples for tool-assisted mashup development

Yahoo Pipes : mix data feeds to create data mashups using a visual editor.

Generated by Targeteam

Mashups

Definition: Mashup simply indicates a way to create new Web applications by combining existing Web resources utilizing data and Web APIs.

Mashup Techniques

Work for the combination of data and services can be done on the server, the client or both of them.

1. Mashing on the Web Server
2. Mashing using Ajax
3. Mashing with JSON

Development Support

Yahoo Pipes are hosted and executed on a Yahoo server.

QedWiki was a Wiki-based mashup maker by IBM; pages are hosted on an IBM server; mostly executed on the client side.

ProgrammableWeb provides a mashup directory and marketplace which let users rank and discuss mashups.

Generated by Targeteam

Distributed Applications - Verteilte Anwendungen

- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching

    Email: schlichter@in.tum.de

    Tel.: 089-289 18654

    URL: http://www11.in.tum.de/

Issues

Software engineering of ***distributed applications*** raises interesting issues. In particular, the following problems must be considered:

1. Specification of a suitable software structure

   Applications must be decomposed into smaller, distributable components; encapsulation of data and functions.

   Which functionality is provided locally and which remotely?

   How should we test and debug distributed applications?

2. Mechanisms for name resolution

   How can an application localize and make use of a remotely provided service?

   Assignment of names to addresses.

   What should happen if a client cannot contact the localized server subsystem?

3. Communication mechanisms

   Selection of the desired communication model, e.g. client-server model, group communication or peer-to-peer.

   How does the application (both client and server) handle network communication errors?

4. Consistency

   How can the data be kept consistent, particularly for replicated data?

Issues

2. Mechanisms for name resolution

   How can an application localize and make use of a remotely provided service?

   Assignment of names to addresses.

   What should happen if a client cannot contact the localized server subsystem?

3. Communication mechanisms

   Selection of the desired communication model, e.g. client-server model, group communication or peer-to-peer.

   How does the application (both client and server) handle network communication errors?

4. Consistency

   How can the data be kept consistent, particularly for replicated data?

   If a cache is used for performance improvement, then it must be kept consistent with the stored data.

   User interface consistency for the individual components.

5. User requirements
   - Functionality and reconfigurability of the distributed application and its components.
   - Service quality, such as security, reliability, fault tolerance and performance.
   - What kind of security mechanisms are provided? Is authentication an issue?

Issues

   What should happen if a client cannot contact the localized server subsystem?

3. Communication mechanisms

   Selection of the desired communication model, e.g. client-server model, group communication or peer-to-peer.

   How does the application (both client and server) handle network communication errors?

4. Consistency

   How can the data be kept consistent, particularly for replicated data?

   If a cache is used for performance improvement, then it must be kept consistent with the stored data.

   User interface consistency for the individual components.

5. User requirements
   - Functionality and reconfigurability of the distributed application and its components.
   - Service quality, such as security, reliability, fault tolerance and performance.
   - What kind of security mechanisms are provided? Is authentication an issue?
   - Which actions will be triggered if a client cannot communicate with its server?
   - What type of heterogeneity is necessary?
   - What efficiency (performance) is expected?

Design of distributed applications

Generated by Targeteam