RESTful Services and Service Engineering

Idaho State University | Computer Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

ROAR

Topics Covered

- RESTful Services
- Service Engineering

RESTful web services

- Current web services standards have been criticized as ‘heavyweight’ standards that are over-general and inefficient.
- REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client.
- This style underlies the web as a whole and is simpler than SOAP/WSDL for implementing web services.
- RESTful services involve a lower overhead than so-called ‘big web services’ and are used by many organizations implementing service-based systems.

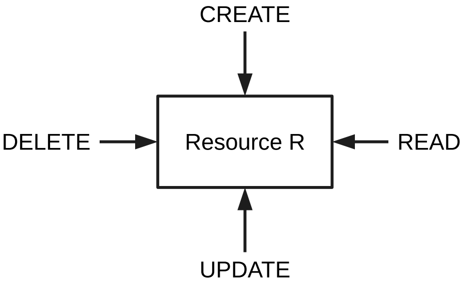
Resources

- The fundamental element in a RESTful architecture is a resource.
- Essentially, a resource is simply a data element such as a catalog, a medical record, or a document, such as this book chapter.
- In general, resources may have multiple representations i.e. they can exist in different formats.
  - MS WORD
  - PDF
  - Quark XPress

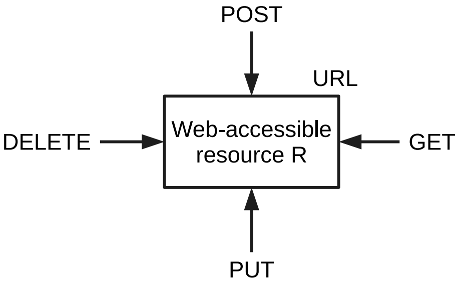
Resource operations

- **Create** – bring the resource into existence.
- **Read** – return a representation of the resource.
- **Update** – change the value of the resource.
- **Delete** – make the resource inaccessible.

Resources and actions



General resource actions



Web resources

Operation functionality

- **POST** is used to create a resource. It has associated data that defines the resource.
- **GET** is used to read the value of a resource and return that to the requestor in the specified representation, such as XHTML, that can be rendered in a web browser.
- **PUT** is used to update the value of a resource.
- **DELETE** is used to delete the resource.

Resource access

- When a RESTful approach is used, the data is exposed and is accessed using its URL.
- Therefore, the weather data for each place in the database, might be accessed using URLs such as:
  - http://weather-info-example.net/temperatures/boston
  - http://weather-info-example.net/temperatures/edinburgh
- Invokes the GET operation and returns a list of maximum and minimum temperatures.
- To request the temperatures for a specific date, a URL query is used:
  - http://weather-info-example.net/temperatures/edinburgh?date=20140226

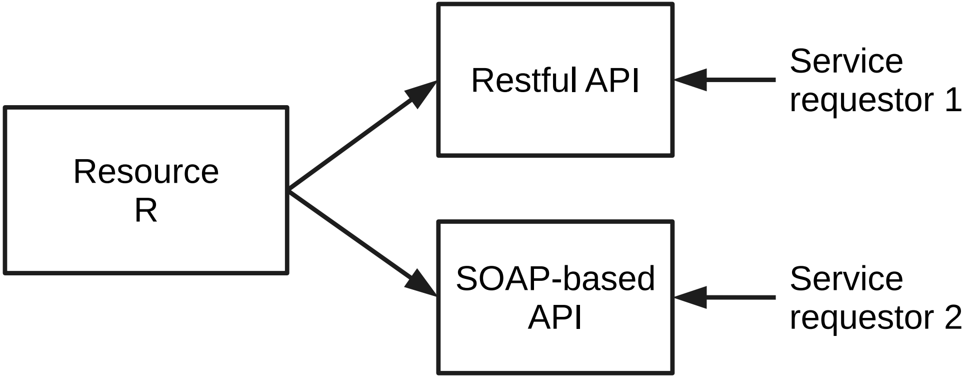
Query results

- The response to a GET request in a RESTful service may include URLs.
- If the response to a request is a set of resources, then the URL of each of these may be included.
  - http://weather-info-example.net/temperatures/edinburgh-scotland
  - http://weather-info-example.net/temperatures/edinburgh-australia
  - http://weather-info-example.net/temperatures/edinburgh-maryland

Disadvantages of RESTful approach

- When a service has a complex interface and is not a simple resource, it can be difficult to design a set of RESTful services to represent this.
- There are no standards for RESTful interface description so service users must rely on informal documentation to understand the interface.
- When you use RESTful services, you have to implement your own infrastructure for monitoring and managing the quality of service and the service reliability.

RESTful and SOAP-based APIs



Resource
R

Restful API

SOAP-based
API

Service
requestor 1

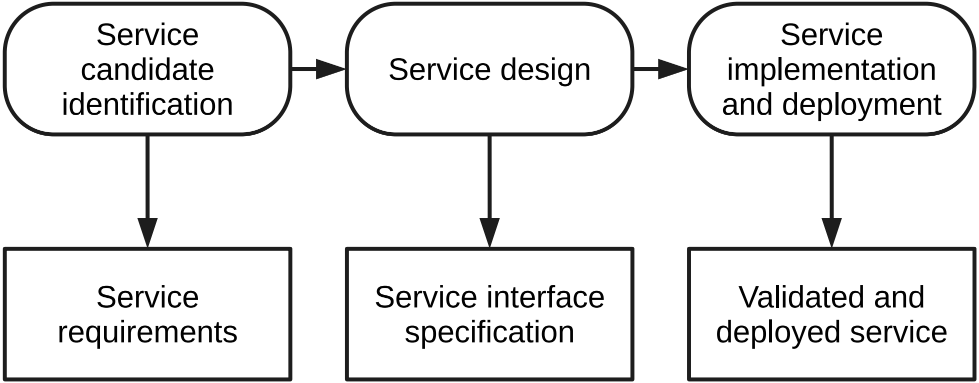
Service
requestor 2

Service engineering

Service engineering

- The process of developing services for reuse in service-oriented applications
- The service has to be designed as a reusable abstraction that can be used in different systems.
- Generally useful functionality associated with that abstraction must be designed and the service must be robust and reliable.
- The service must be documented so that it can be discovered and understood by potential users.

The service engineering process



Service candidate identification

Service design

Service implementation and deployment

Service requirements

Service interface specification

Validated and deployed service

Stages of service engineering

- Service candidate identification, where you identify possible services that might be implemented and define the service requirements.
- Service design, where you design the logical service interface and its implementation interfaces (SOAP and/or RESTful)
- Service implementation and deployment, where you implement and test the service and make it available for use.

Service candidate identification

- Services should support business processes.
- Service candidate identification involves understanding an organization's business processes to decide which reusable services could support these processes.
- Three fundamental types of service
  - Utility services that implement general functionality used by different business processes.
  - Business services that are associated with a specific business function e.g., in a university, student registration.
  - Coordination services that support composite processes such as ordering.

Task and entity-oriented services

- Task-oriented services are those associated with some activity.
- Entity-oriented services are like objects. They are associated with a business entity such as a job application form.
- Utility or business services may be entity- or task-oriented, coordination services are always task-oriented.

Service classification

	Utility	Business	Coordination
Task	Currency converter	Validate claim form	Process expense claim
	Employee locator	Check credit rating	Pay external supplier
Entity	document style checker	Expenses form	
	Web form to XML converter	Student application form	

Service identification

- Is the service associated with a single logical entity used in different business processes?
- Is the task one that is carried out by different people in the organization? Can this fit with a RESTful model?
- Is the service independent?
- Does the service have to maintain state? Is a database required?
- Could the service be used by clients outside the organization?
- Are different users of the service likely to have different non-functional requirements?

Service identification example

- A large company, which sells computer equipment, has arranged special prices for approved configurations for some customers.
- To facilitate automated ordering, the company wishes to produce a catalog service that will allow customers to select the equipment that they need.
- Unlike a consumer catalog, orders are not placed directly through a catalog interface. Instead, goods are ordered through the web-based procurement system of each company that accesses the catalog as a web service.
- Most companies have their own budgeting and approval procedures for orders and their own ordering process must be followed when an order is placed.

Catalog services

- Created by a supplier to show which good can be ordered from them by other companies
- Service requirements
  - Specific version of catalog should be created for each client
  - Catalog shall be downloadable
  - The specification and prices of up to 6 items may be compared
  - Browsing and searching facilities shall be provided
  - A function shall be provided that allows the delivery date for ordered items to be predicted
  - Virtual orders shall be supported which reserve the goods for 48 hours to allow a company order to be placed

Non-functional requirements

- Access shall be restricted to employees of accredited organizations
- Prices and configurations offered to each organization shall be confidential
- The catalog shall be available from 0700 to 1100
- The catalog shall be able to process up to 10 requests per second

Catalog service operations

Operation	Description
MakeCatalog	Creates a version of the catalog tailored for a specific customer. Includes an optional parameter to create a downloadable PDF version of the catalog.
Lookup	Displays all of the data associated with a specified catalog item.
Search	This operation takes a logical expression and searches the catalog according to that expression. It displays a list of all items that match the search expression.

Catalog service operations

Operation	Description
Compare	Provides a comparison of up to six characteristics (e.g., price, dimensions, processor speed, etc.) of up to four catalog items.
CheckDelivery	Returns the predicted delivery date for an item if ordered that day.
MakeVirtualOrder	Reserves the number of items to be ordered by a customer and provides item information for the customer’s own procurement system.

Service interface design

- Involves thinking about the operations associated with the service and the messages exchanged
- The number of messages exchanged to complete a service request should normally be minimized.
- Service state information may have to be included in messages

Interface design stages

- Logical interface design
  - Starts with the service requirements and defines the operation names and parameters associated with the service. Exceptions should also be defined
- Message design (SOAP)
  - For SOAP-based services, design the structure and organization of the input and output messages. Notations such as the UML are a more abstract representation than XML
  - The logical specification is converted to a WSDL description
- Interface design (REST)
  - Design how the required operations map onto REST operations and what resources are required.

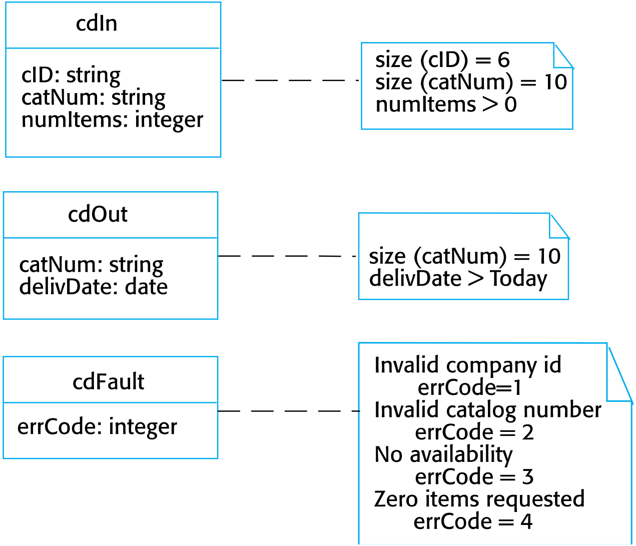
Catalog interface design

Operation	Inputs	Outputs	Exceptions
MakeCatalog	mcIn Company id PDF-flag	mcOut URL of the catalog for that company	mcFault Invalid company id
Lookup	lookIn Catalog URL Catalog number	lookOut URL of page with the item information	lookFault Invalid catalog number
Search	searchIn Catalog URL Search string	searchOut URL of web page with search results	searchFault Badly formed search string

Catalog interface design

Operation	Inputs	Outputs	Exceptions
Compare	compIn Catalog URL Entry attribute (up to 6) Catalog number (up to 4)	compOut URL of page showing comparison table	compFault Invalid company id Invalid catalog number Unknown attribute
CheckDelivery	cdIn Company id Catalog number Number of items required	cdOut Catalog number Expected delivery date	cdFault Invalid company id No availability Zero items requested
MakeVirtualOrder	poIn Company id Number of items required Catalog number	poOut Catalog number Number of items required Predicted delivery date	poFault Invalid company id Invalid catalog number Zero items requested

Input and output message



cdIn

cID: string
catNum: string
numItems: integer

size (cID) = 6
size (catNum) = 10
numItems > 0

cdOut

catNum: string
delivDate: date

size (catNum) = 10
delivDate > Today

cdFault

errCode: integer

Invalid company id
errCode=1
Invalid catalog number
errCode = 2
No availability
errCode = 3
Zero items requested
errCode = 4

RESTful interface

- There should be a resource representing a company-specific catalog. This should have a URL of the form `<base catalog>/<company name>` and should be created using a POST operation.
- Each catalog item should have its own URL of the form:
  - `<base catalog>/<company name>/<item identifier>`
- The GET operation is used to retrieve items.
  - **Lookup** is implemented by using the URL of an item in a catalog as the GET parameter.
  - **Search** is implemented by using GET with the company catalog as the URL and the search string as a query parameter. This GET operation returns a list of URLs of the items matching the search.

RESTful interface

- The **Compare** operation can be implemented as a sequence of GET operations, to retrieve the individual items, followed by a POST operation to create the comparison table and a final GET operation to return this to the user.
- The **CheckDelivery** and **MakeVirtualOrder** operations require an additional resource, representing a virtual order.
  - A POST operation is used to create this resource with the number of items required. The company id is used to automatically fill in the order form and the delivery date is calculated. This can then be retrieved using a GET operation.

Service implementation and deployment

- Programming services using a standard programming language or a workflow language
- Services then have to be tested by creating input messages and checking that the output messages produced are as expected
- Deployment involves publicizing the service and installing it on a web server. Current servers provide support for service installation

Legacy system services

- Services can be implemented by implementing a service interface to existing legacy systems
- Legacy systems offer extensive functionality and this can reduce the cost of service implementation
- External applications can access this functionality through the service interfaces

Service descriptions

- Information about your business, contact details, etc. This is important for trust reasons. Users of a service have to be confident that it will not behave maliciously.
- An informal description of the functionality provided by the service. This helps potential users to decide if the service is what they want.
- A description of how to use the services SOAP-based and RESTful.
- Subscription information that allows users to register for information about updates to the service.

Key points

- Alternatively, a RESTful architecture may be used which is based on resources and standard operations on these resources.
- A RESTful approach uses the http and https protocols for service communication and maps operations on the standard http verbs POST, GET, PUT and DELETE.
- Utility services provide general-purpose functionality; business services implement part of a business process; coordination services coordinate service execution.
- Service engineering involves identifying candidate services for implementation, defining service interfaces and implementing, testing and deploying services.

Are there any questions?