Copyright Notice

This presentation is intended to be used only by the participants who attended the RESTful web services with java training session by Prakash Badhe for CTS.
Sharing/selling of this presentation in any form is NOT permitted.
Others found using this presentation or violation Of above terms is considered as legal offence.

AgileSoft Methodologies

REST Ful Web Services

Prakash Badhe
Prakash.badhe@agilesoft.in

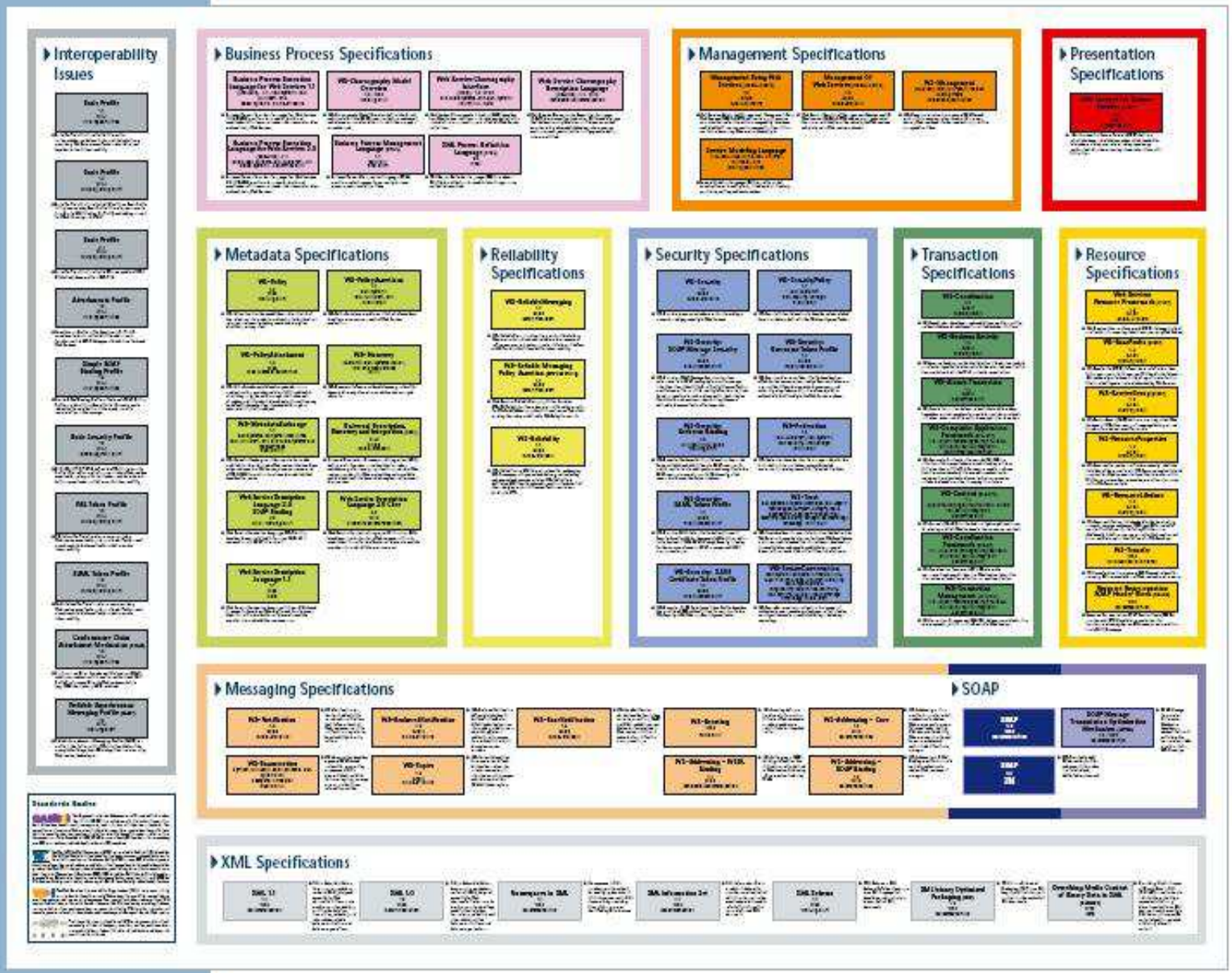
Soap Web Services

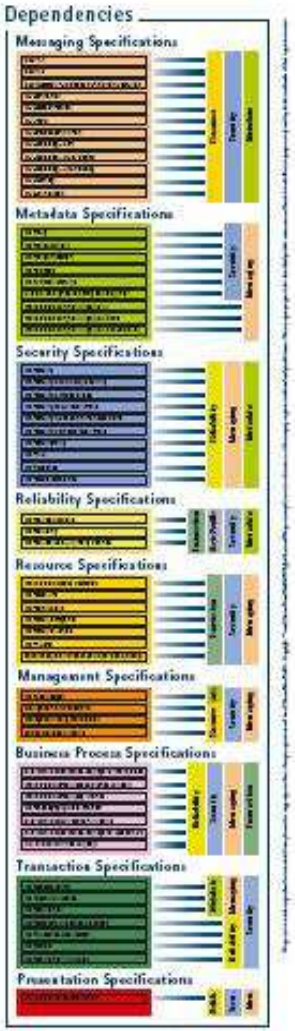
- Manage language independent interactions across different platforms and applications
- Different versions evolved one by one to support additional features.
- Interoperability is still an issue across different implementations and versions.
- Heavy dependencies on the soap library and other xml implementations.
- Non-xml data format support is limited only in the form of attachments.
- Performance is an issue for mission critical applications.

What is wrong with SOAP Services ?

- Language first mindset
- WSDL imposes tightly coupled dependencies between services and consumers
- No support for dynamic and functional languages
- With a few exceptions: WS protocols are not ready for the internet
- We have too many WS protocols and too many versions

Web Services Standards Overview





Version 3.0 - February 2007

RESTing the SOAP ?

Need of REST

- We need a natural way to model resource-based services
- We should leverage the principles of the web on SOA applications

AgileSoft Methodologies

Resource Oriented Architecture

- ROA is the term for REST on HTTP/URI
- A Service consists of all the resources available within a certain domain of control
- Since REST is a type of SOA, ROA is an implementation of SOA as well.

AgileSoft Methodologies

REST Architecture

- Representational State Transfer (REST) is an architectural style that specifies the uniform interface that applied to a web resource and supports performance, scalability, and modifiability that enable services to work best on the Web.
- In REST architectural style, data and functionality are considered resources and are accessed using *Uniform Resource Identifiers (URIs)*, typically by links on the Web.
- REST works with resources addressable on the web typically by links.

REST History

- Fielding captured his interpretation of the www architecture in his 2000 thesis
- Championed by the Web community
- Positioned as an alternative to WS-* implementations
- Adopted by a few major vendors (Microsoft:WCF, Sun:Jersey)
- High impact in the community

REST Actions

- The resources are acted upon by using a set of simple, well-defined operations.
- The REST architectural style specifies a client/server architecture and is designed to use a stateless communication protocol, typically HTTP.
- In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.
- In REST architecture, http methods are used as operations on resources.

‘REST’ Services

- REST services work with http in web applications.
- REST is an architectural style rather than a protocol which removes the dependencies on soap and xml standards as wsdl,uddi etc.
- REST supports any understandable data formats across applications.
- REST specifies Resources/data on server rather than actions on them..
- REST specifies transfer of the state of Resource across applications.

REST vs SOAP Battle

AgileSoft Methodologies

REST vs SOAP

- REST is ready for the enterprise
- REST is strong at:
  - Internet scale computing
  - High levels of interoperability
  - Resource Oriented operations
- SOAP/WS is strong at:
  - Complex security (Trust and Federation)
  - Multi-transport services
  - Occasionally connected applications
- In the real world they are typically enabled by a combination of WS and REST

REST Principles

Resource based (not service based)

Addressability (name everything that matters)

Statelessness (no stateful messages exchanged with a resource)

Relationships (expressed through links)

State Machine interactions (not business processes)

HTTP based

Http based

- REST principles are not HTTP dependent
- But in practice now REST is be considered an HTTP dependent architecture style
- REST expresses dependencies on HTTP specific concepts such as verbs, URIs and headers
- In the future....non HTTP-based REST will come up..

AgileSoft Methodologies

What is Resource ?

- A resource is something “interesting” in the system
- Can be anything
  - Spreadsheet (or one of its cells)
  - Blog posting
  - Printer
  - Winning lottery numbers
  - A transaction
  - Others?
- Making your system Web-friendly increases its surface area
  - You expose many **resources**, rather than fewer *endpoints*

Resources on the Web

Resources are not only represented as XML

- XML formats (HTML, XHTML, RSS, etc.)
- JPG, GIF, PNG
- MP3, WAV, OGG
- Anything else that can be on the web.

AgileSoft Methodologies

Resource Nouns

- Important ‘things’ (nouns) are Resources
  - Addressed through a URI
- Uniform interface (verbs)
  - In HTTP: GET, PUT, POST, DELETE
- Verb-noun seperation makes intégration easier
  - GET /customer/45 Instead of getCustomer(45)

AgileSoft Methodologies

REST verbs (operations)

- REST works with resources as nouns with their identities.
- The state of these nouns is shared with clients over http methods
- The http methods are specified as verbs in REST
  - Get: get the resource state/values
  - Post : post new resource
  - Put : update the resource state
  - Delete :delete the resource on server
  - Options: read the options available with server.
  - Head : set the http headers on server.

AgileSoft Methodologies

REST Commandments

- Give every “thing” an ID
- Link things together
- Use standard methods
- Communicate statelessly

AgileSoft Methodologies

Addressability of resources

- Resources MUST be represented by URIs
- Retrieve a representation of a resource: GET
- Get metadata about an existing resource: HEAD
- Create a new resource: PUT to a new URI, or POST to an existing URI
- Modify an existing resource: PUT to an existing URI
- Delete an existing resource: DELETE
- See which of the verbs the resource understands: OPTIONS

Resource Links

- Resources contain links (or URI templates) to other resources
- Links act as state transitions
- Think of resources as states in a state machine
- And links as state transitions
- Application (conversation) state is captured in terms of these states

AgileSoft Methodologies

Configuration Annotations

- The Java programming annotations defined by JAX-RS JSR-311 specification.
- @Path: specifies the relative URI path indicating where the Java class will be hosted on the server.
- @GET: resource method mapped to the HTTP request get method.
- @POST, @PUT,@ DELETE, @HEAD are mapped.

AgileSoft Methodologies

Extract Request Parameters

- The parameters of a resource method are annotated with parameter-based annotations to extract information from a request.
- @PathParam parameter to extract a path parameter from the path component of the request URL that matched the path declared in @Path.

AgileSoft Methodologies

Request params

- To extract the types of parameters for use in the resource class
  - Query
  - URI path
  - Form
  - Cookie
  - Header
  - Matrix

AgileSoft Methodologies

Request parameters

- @PathParam: specifies URI path parameters to be extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation.
- @QueryParam: The query parameters are extracted from the request URI query parameters and are injected in the variables.
- @MatrixParam, @HeaderParam, @CookieParam, and @FormParam follow the same rules as @QueryParam.

Produce response

- @Produces : used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain“, “application/xml”, “application/json” etc.
- @Consumes: specifies the MIME media types of representations a resource can consume that were sent by the client. The request media types are same as above in produces.
- @Provider: Configures the JAX-RS runtime as MessageBodyReader and MessageBodyWriter.

AgileSoft Methodologies

Reader and Writer

- For HTTP requests, the MessageBodyReader is used to map an HTTP request entity body to method parameters.
- On the response side, a return value is mapped to an HTTP response entity body by using a MessageBodyWriter.
- If the application needs to supply additional metadata, such as HTTP headers or a different status code, a method can return a Response that wraps the entity and that can be built using Response.ResponseBuilder.

Statelessness

- The interactions with resources are stateless
- Resource state is always kept in the server and sent to the client as representations
- Application state is always kept in the client and is used to modify the state of resources
- Statelessness increments scalability (self contained messages) ..see the overhead of managing Stateful EJB..
- Avoids putting resources in inconsistent state

State management

- REST mandates that state be either turned into resource state, or kept on the client.
- Reduces burden on the server..increases scalability on the server
- Reduces coupling with a specific machine

AgileSoft Methodologies

Java support for REST

- JAX-RS JSR-311 is the new java standard for implementing and consuming REST web services.
- Implementations are Jboss RestEasy,Jersey, RESTlet platforms.

AgileSoft Methodologies

Resource Locators and Sub Resources

- Resource classes are able to partially process a request and provide another "sub" resource object that can process the remainder of the request.

AgileSoft Methodologies

Sub-resource Locator

- Resource methods that have a @Path annotation, but no HTTP method marked are considered sub-resource locators.
- Their job is to provide an object that can process the request.
- In the above example ShoppingStore is a root resource because its class is annotated with @Path. The getCustomer() method is a sub-resource locator method.

Sub-resource locator


```
• @Path("/") public class ShoppingStore {
• @Path("/customers/{id}")
• public Customer getCustomer(@PathParam("id") int id) {
•   Customer cust = ...; // Find a customer object
•   return cust;
•   } }
•  public class Customer {
• @GET public String get() {...}
• @Path("/address")
•  public String getAddress() {
• ..} }
```


AgileSoft Methodologies

Resource Locator Usage

- If the client invoked:
- GET /customer/12
- The ShoppingStore.getCustomer() method would be invoked first and returns a Customer object that can service the request.
- The http request will be dispatched to the Customer.get() method.
- GET /customer/11/address
- ShoppingStore.getCustomer() method is invoked. A customer object is returned, and the rest of the request is dispatched to the Customer.getAddress() method.

Header Manipulation

- Using a filter in REST application
- Using an Interceptor MessageBodyWriterInterceptor and PostProcessInterceptor
- Using Response object

AgileSoft Methodologies

Dynamic Locators

- The locator method result is dynamically processed at runtime to figure out how to dispatch the request.
- T ShoppingStore.getCustomer() method does not have to declare any specific type.
- It can return different types of objects and hence different path at runtime.
- @Path("/customers/{id}")

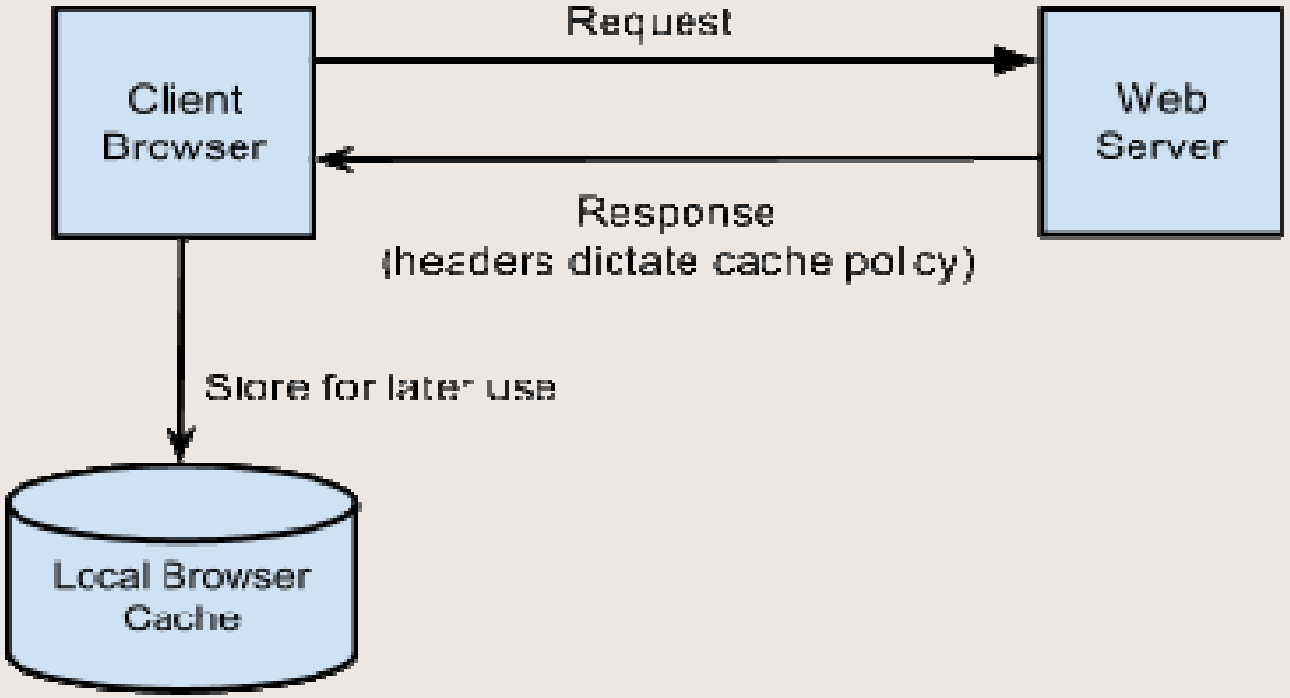
```
public java.lang.Object getCustomer(@PathParam("id") int id){
//Hibernate session to load db object
 Customer cust = session.find(Customer.class, id);
 return cust;
}
```


AgileSoft Methodologies

Browser cache

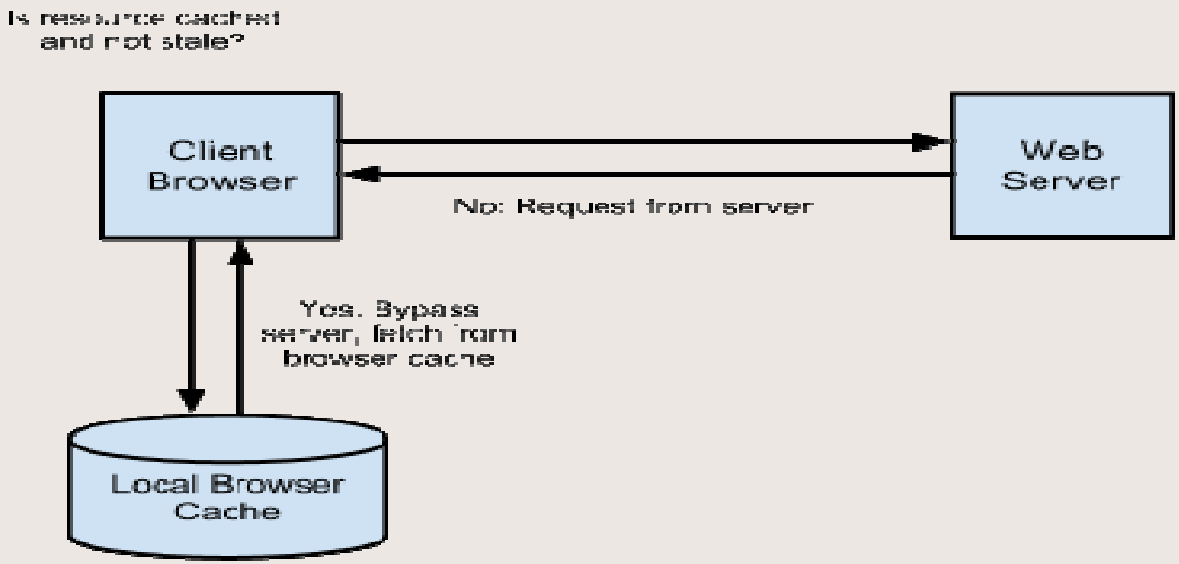
- HTTP caching is an easy option to speed up web applications.
- It's standardized and well implemented in all modern browsers and results in a lower latency app and improved responsiveness.
- HTTP caching occurs when the browser stores local copies of web resources for faster retrieval the next time the resource is required.
- As the application serves resources it can attach cache headers to the response specifying the desired cache behavior.

Resource-first time loading



AgileSoft
Methodologies

Next time loading from Cache



When an item is fully cached, the browser may choose to not contact the server at all and simply use its own cached copy:

AgileSoft Methodologies

Caching to improve performance

- As an example once CSS style sheets/images from the application are downloaded by the browser, there's no need to download them again during the user's session.
- This happens for many asset types like JavaScript files, images and even infrequently changing dynamic content.
- In these instances it is beneficial for the users browser to cache this file locally, and use that copy whenever the resource is requested again. An application using HTTP cache headers is able to control this caching behavior and alleviate server-side load.

Http Cache Header

- There are two primary cache headers,
  - Cache-Control
  - Expires.
- The Cache-Control header effectively ‘switches on’ caching in the browser.
- With this header in place, and set with a value that enables caching, the browser will cache the file for as long as specified. Without this header the browser will re-request the file on each subsequent request.

Intermediate Node Cache

- public resources can be cached not only by the end-user's browser but also by any intermediate proxies that may be serving many other users as well.
- Cache-Control:public
- private resources are bypassed by intermediate proxies and can only be cached by the end-client.
- Cache-Control:private

AgileSoft Methodologies

Cache Expiry

- The value of the Cache-Control header is a composite, indicating whether the resource is public or private while also indicating the maximum amount of time it can be cached before considered stale.
- The max-age value sets a time-span for how long to cache the resource (in seconds).
- Cache-Control:public, max-age=31536000
- While the Cache-Control header turns on client-side caching and sets the max-age of a resource theExpires header is used to specify a specific point in time the resource is no longer valid.
- Expires: Mon, 25 Jun 2012 21:31:12 GMT

Set the expiry of cache

- The Expires header sets a date from which the cached resource should no longer be considered valid.
- From this date forward the browser will request a fresh copy of the resource. Until then, the browsers local cached copy will be used.
- If both Expires and max-age are set max-age will take precedence.

AgileSoft Methodologies

Response Cache Header

- HTTP/1.1 200 OK
- Content-Type: application/json
- Cache-Control: private, max-age=86400 Last-Modified: Thu, 07 Feb 2013 11:56 EST
- This response would be cached for one day

AgileSoft Methodologies

Conditional Requests

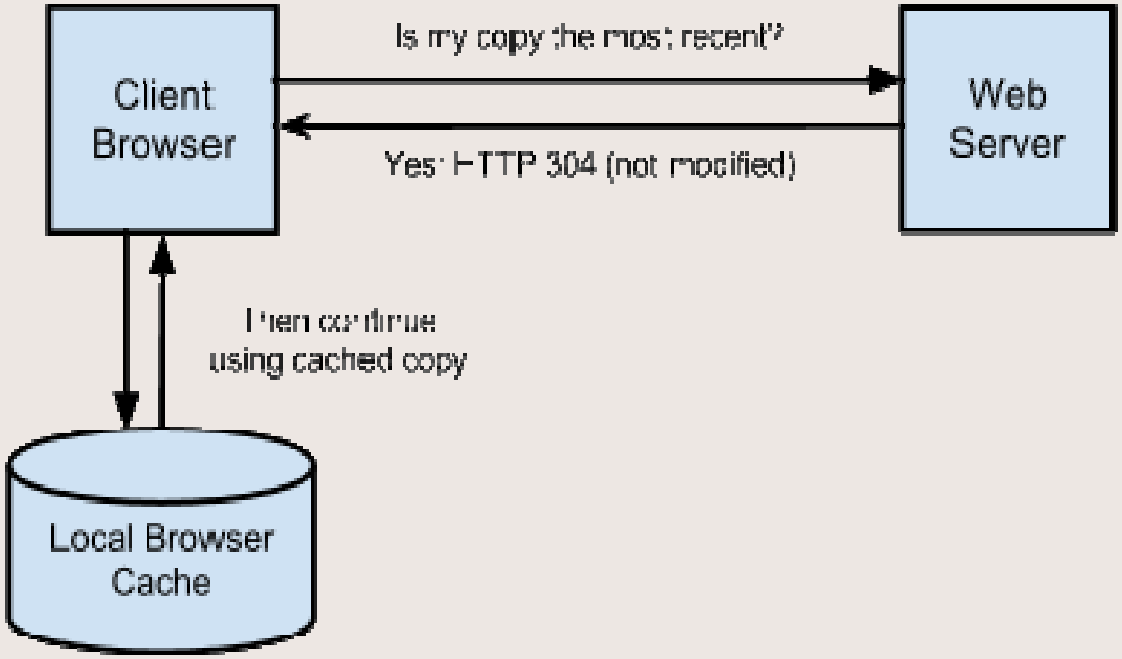
- The Cache-Control and Expires header tells the browser *when* to next retrieve the resource from the network .
- A few additional headers specify *how* to retrieve the resource from the network. These types of requests are known as *conditional requests*.

AgileSoft Methodologies

Resource Modified..?

- Conditional requests are those where the browser requests/queries the server if it has an updated copy of the resource.
- The browser sends some information about the cached resource it holds and the server determines whether updated content should be returned or the browser's copy is the most recent.
- In the case of the latter an HTTP status of 304 (not modified) is returned.

Resource Query..



AgileSoft Methodologies

Saving the loading time

- The conditional requests do invoke a call across the network, unmodified resources result in an empty response body – saving the cost of transferring the resource back to the end client.
- The backend service is also often able to very quickly determine a resource's last modified date without accessing the resource which itself saves non-trivial processing time.

AgileSoft Methodologies

Time-based conditional request

- A time-based conditional request ensures that only if the requested resource has changed since the browser's copy was cached will the contents be transferred.
- If the cached copy is the most up-to-date then the server returns the 304 response code.
- To enable conditional requests the application specifies the last modified time of a resource via theLast-Modified response header.

AgileSoft Methodologies

Last Modified Time

- The server returns the header as part of response
- Cache-Control:public, max-age=31536000 Last-Modified: Mon, 03 Jan 2011 17:45:57 GMT

AgileSoft Methodologies

Conditional Request Header

- The next time the browser requests the resource it will only ask for the contents of the resource if they're unchanged since this date using the header *If-Modified-Since request* header.
- If-Modified-Since: Mo
- If the resource hasn't changed since Mon, 03 Jan 2011 17:45:57 GMT the server will return with an empty body with the 304 response code.

AgileSoft Methodologies

Content-based header

- The ETag (or Entity Tag) works in a similar way to the Last-Modified header except its value is a digest of the resources contents (for instance, an MD5 hash).
- This allows the server to identify if the cached contents of the resource are different to the most recent version.
- This tag is useful when for when the last modified date is difficult to determine.
- Cache-Control:public, max-age=31536000
- ETag: "15f0fff99ed5aae4edffdd6496d7131f"

AgileSoft Methodologies

Usage of ETag

- On subsequent browser requests the If-None-Match *request* header is sent with the ETag value of the last requested version of the resource.
- If-None-Match: "15f0fff99ed5aae4edffdd6496d7131f"
- As with the If-Modified-Since header, if the current version has the same ETag value, indicating its value is the same as the browser's cached copy, then an HTTP status of 304 is returned.

Observe Headers

- Most modern browsers include robust request/response visualization and introspection tools.
- The Web Inspector found in both Chrome,FireFox and Safari shows the response and request headers in the ‘Network’ tab.

AgileSoft Methodologies

Disable Cache

- Highly secure or variable resources often require no caching.
- For instance, in a shopping cart checkout process.
- Unfortunately, merely omitting cache headers will not work as many modern web browsers cache items based on their own internal algorithms. In such cases it is necessary to tell the browser to explicitly to not cache items.
- In addition to public and private the Cache-Control header can specify no-cache and no-store which informs the browser to not cache the resources under any circumstances.
- Cache-Control:no-cache, no-store

REST applications cache Usage

- In a JAX-RS REST method a Response object can be returned.
- The content will be auto transformed according to the specified MIME-Type (this is called JAX-RS Content Negotiation).
- The Cache-Control header can also be set on the Response object by using the CacheControl class.
- Methods are provided for all of the available Cache-Control header tokens.

AgileSoft Methodologies

Set Cache Header


```
@Path("/book/{id}")
@GET public Response getBook(@PathParam("id") long id){
   Book myBook = getBookFromDB(id);
   CacheControl cc = new CacheControl();
   cc.setMaxAge(86400);
   cc.setPrivate(true);
    ResponseBuilder builder = Response.ok(myBook);
    builder.cacheControl(cc);
    return builder.build();
}
```

Response Headers

▼ Response Headers view parsed

```
HTTP/1.1 200 OK
X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server
Server: GlassFish Server Open Source Edition 3.1.2.2
Cache-Control: private, no-transform, max-age=86400
Content-Type: application/json
Transfer-Encoding: chunked
Date: Thu, 07 Mar 2013 12:46:42 GMT
```


AgileSoft Methodologies

Conditional cache headers

- The Conditional requests are those where the browser can ask the server if it has an updated copy of the resource.
- The browser will send one or both of the ETag and If-Modified-Since headers about the cached resource it holds.
- The server can then determine whether updated content should be returned or the browser's copy is the most recent.

AgileSoft Methodologies

Conditional GET Request

- In JAX-RS REST application the Request object has a method to check whether the data based on the date or the etag from the client was modified.
- The ETag value should be a unique hash of the resource and requires knowledge of internal business logic to construct.
- One possibility is to use the HashCode of a object as the etag.

AgileSoft Methodologies

Implement Conditional Get


```
@Path("/book/{id}")
@GET
public Response getBook(@PathParam("id") long id, @Context Request
    request){
Book myBook = getBookFromDB(id);
CacheControl cc = new CacheControl();
cc.setMaxAge(86400);
EntityTag etag = new EntityTag(Integer.toString(myBook.hashCode()));
ResponseBuilder builder = request.evaluatePreconditions(etag);
// cached resource did change -> serve updated content
 if(builder == null){
builder = Response.ok(myBook); builder.tag(etag); }
builder.cacheControl(cc); return builder.build();
 }
```


AgileSoft Methodologies

Conditional Get Response

A ResponseBuilder is automatically constructed by calling evaluatePreconditions on the request. If the builder is null, the resource is out of date and needs to be sent back in the response. Otherwise, the preconditions indicate that the client has the latest version of the resource and the 304 Not Modified status code will be automatically assigned.

```
▼Response Headers    view parsed
  HTTP/1.1 200 OK
  X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server
  Server: GlassFish Server Open Source Edition 3.1.2.2
  ETag: "192010067"
  Cache-Control: no-transform, max-age=86400
  Content-Type: application/json
  Transfer-Encoding: chunked
  Date: Thu, 07 Mar 2013 12:46:08 GMT
```

```
▼Response Headers    view parsed
  HTTP/1.1 304 Not Modified
  X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server
  Server: GlassFish Server Open Source Edition 3.1.2.2
  ETag: "192010067"
  Cache-Control: no-transform, max-age=86400
  Date: Thu, 07 Mar 2013 12:45:16 GMT
```

Conditional PUT

- The resource updates can also be conditional.
- The client would receive 412 Precondition Failed if theETag send by the client does not match the one on the server.
- Otherwise the update could be made and a 200 OK with the updated entity or a 204 No Content would be send back.

AgileSoft Methodologies

Conditional PUT Code


```
@Path("/book/{id}")
@PUT @Consumes("application/json")
public Response getBook(@PathParam("id") long id, @Context Request
    request, Book updatedBook){
 Book myBook = getBookFromDB(id); EntityTag etag = new
EntityTag(Integer.toString(myBook.hashCode()));
ResponseBuilder builder = request.evaluatePreconditions(etag);
// client is not up to date (send back 412) if(builder != null){
return builder.build(); }
updateBookInDB(updatedBook);
builder = Response.noContent();
return builder.build(); }
```


AgileSoft Methodologies

Conditional PUT Response Headers

▼Response Headers view parsed

```
HTTP/1.1 204 No Content
X-Powered-By: Servlet/3.0 JSP/2.2 (GlassFish Server
Server: GlassFish Server Open Source Edition 3.1.2.2
Date: Fri, 08 Mar 2013 09:02:13 GMT
```


AgileSoft Methodologies

REST for Mobile clients

- By using REST over HTTP instead of SOAP, we can drastically reduce the overhead of message processing
- An HTTP server implementation is feasible for mobile devices

http://opensource.nokia.com/projects/mobile-web-server/

AgileSoft Methodologies

REST on the WEB

- Yahoo! : Weather services
- Google
  - GData, OpenSocial
- Standards
  - Atom, WebDAV
- Amazon
  - S3, SimpleDB
- Microsoft (!)
  - Project Astoria, Web3S
- Um… the Web.

AgileSoft Methodologies

Resources on Social Web

- Users consume content by a lot of sources, in many formats. (Web, Feeds, YouTube, Amazon, Flickr, etc.)
- Users share things they find interesting by exposing a Feed, through Google Reader, Facebook, and FriendFeed.
- The people who read the feeds may share the content with their contacts as well.

AgileSoft Methodologies

ATOM protocol and resource

- Atom Syndication Format
  - A Feed specification based on RSS 2.0
- Atom Publishing Protocol
  - a simple HTTP-based protocol for creating and updating web resources.

AgileSoft Methodologies

Google OpenSocial

- Open Competitor to Facebook's Application Platform
- Exposes Three RESTful APIs:
  - People and Friends data API
  - Activities data API
  - Persistence data API

AgileSoft Methodologies

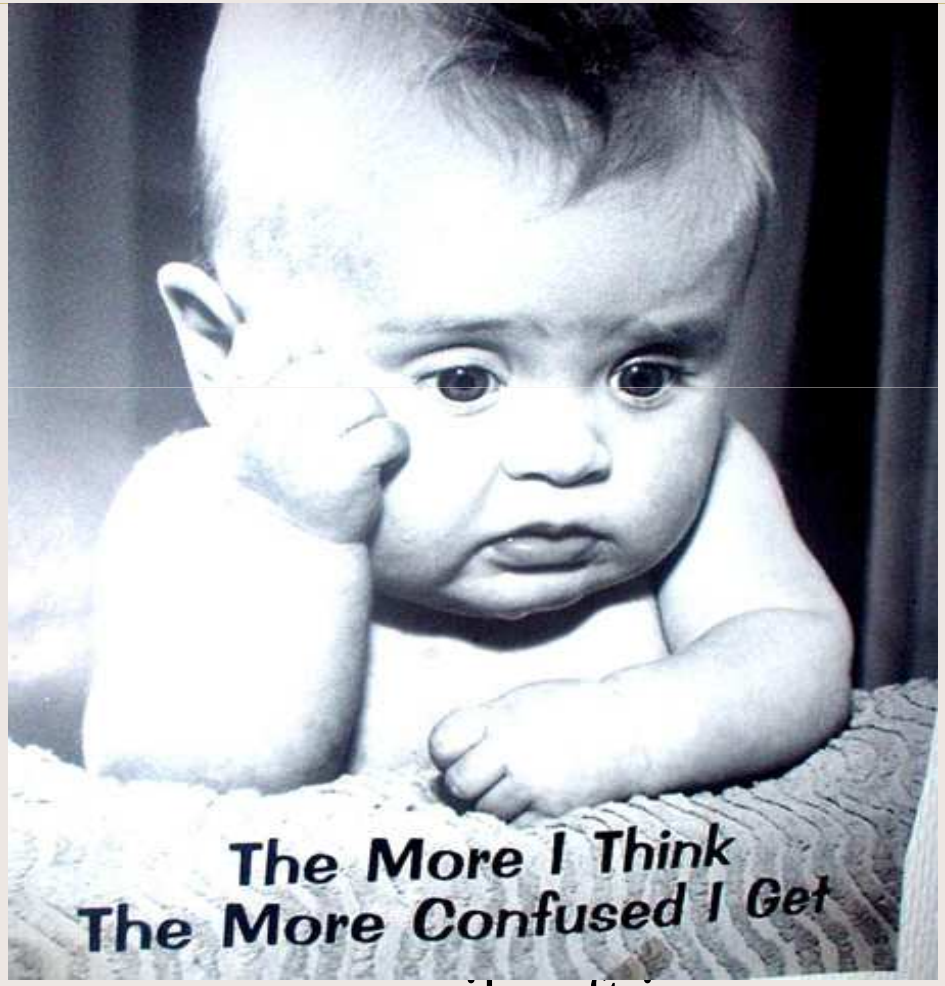
Constraints for the REST

REST has four architectural constraints:

- separation of resource from representation,
- uniform interface,
- self-descriptive messages, and
- hypermedia as the engine of application state.

AgileSoft Methodologies

Questions ?



AgileSoft Methodologies

Thank You!

www.agilesoft.in

AgileSoft Methodologies