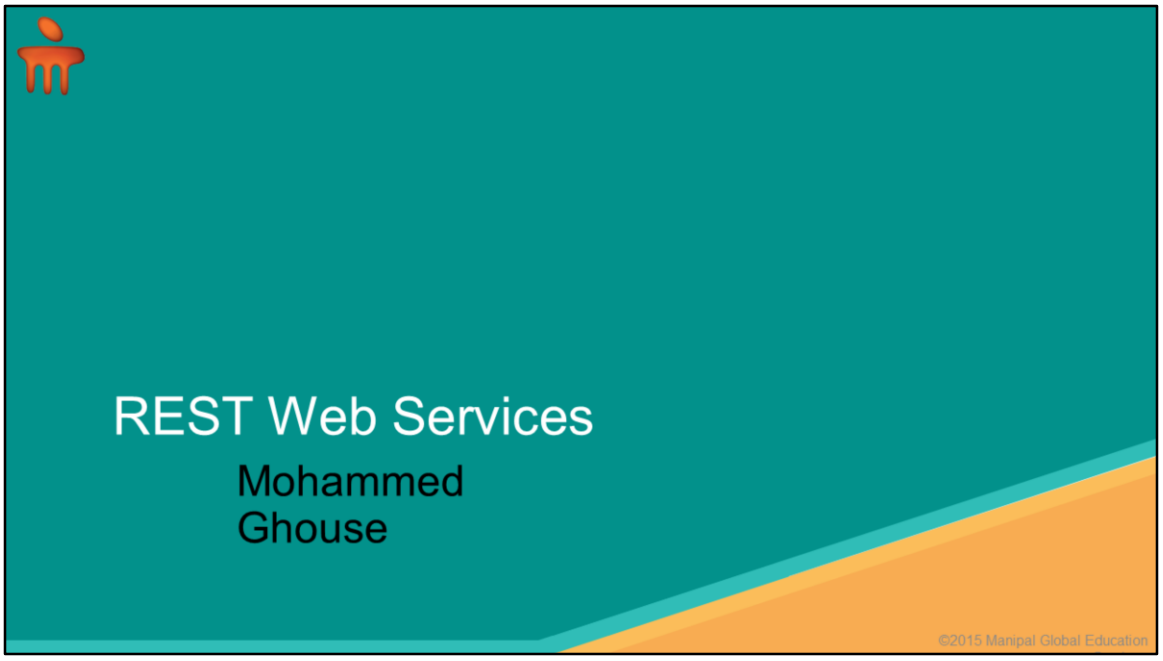

REST Web Services
Mohammed
Ghouse
©2015 Manipal Global Education

manipalglobal
education services

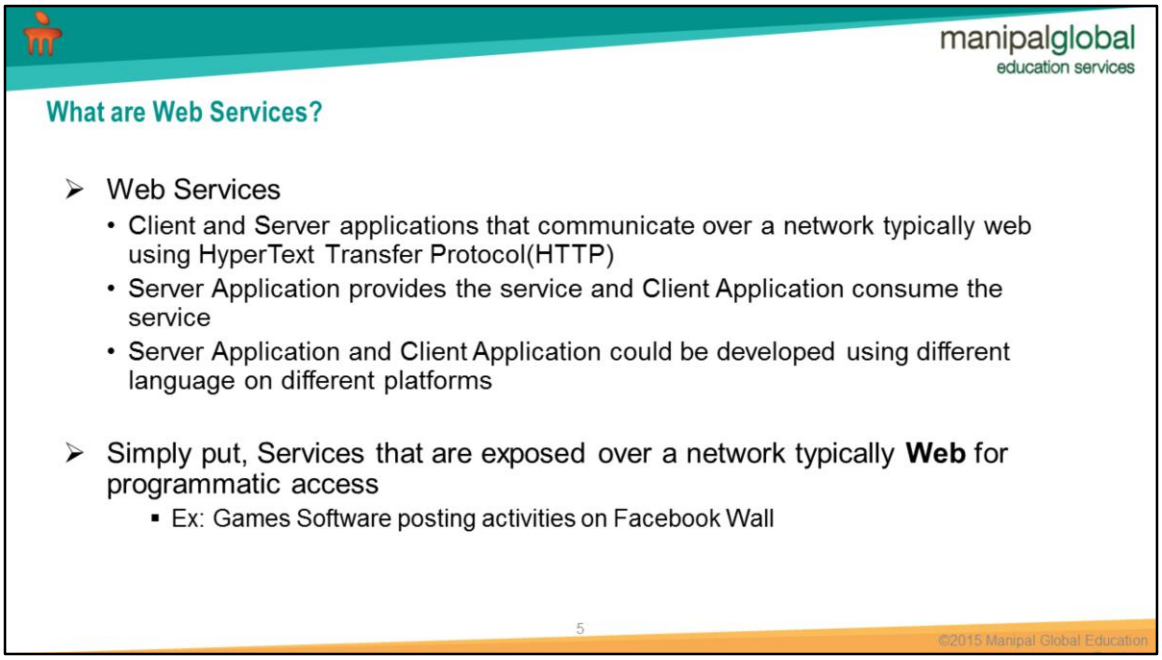
LEARNING OBJECTIVES

At the end of this lesson, you will be able to:

- Understand web services and its types
- Describe REST architectural principles
- Explain REST web services
- Design REST API
- Build REST web services using JAX-RS
- Invoke REST web services using JAX-RS client


manipalglobal
education services
Refer project **REST_WebService** in the provided code base for demo programs on the topics covered in this presentation
3
©2015 Manipal Global Education


manipalglobal
education services
CONCEPT
Web Services
Overview
4
©2015 Manipal Global Education



Modern day business applications use variety of programming platforms to develop web-based applications. Some applications may be developed in Java, others in .Net, while some other in Angular JS, Node.js, etc. Most often than not, these heterogeneous applications need some sort of communication to happen between them. Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other over a network typically internet.

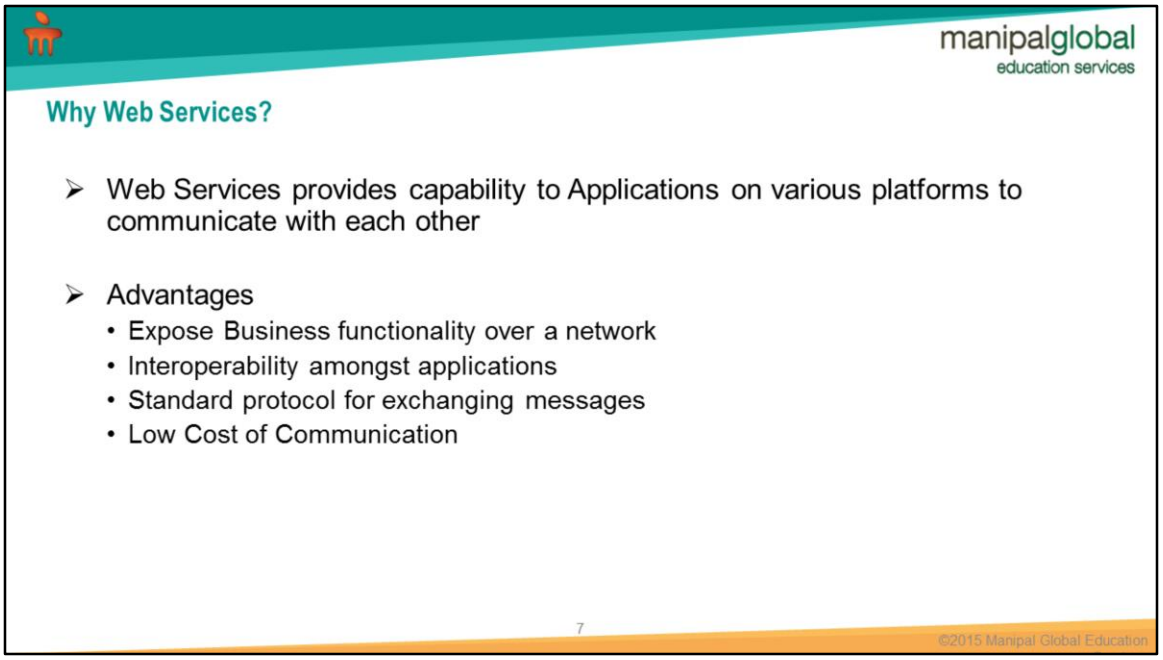
Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticatedSadded-value services.

manipalglobal
education services

Web Services vs Web Application

Web applications are also accessible over the network and provide various kind of services. So, How are Web Services different?

Web Applications	Web Services
➢ Consumption by humans	➢ Consumptions by Applications
➢ Accessed only through a browser	➢ Accessed by applications built on a platform independent of the Web service provider platform
➢ Data sent by Web applications is presented in human readable format	➢ Data exchange happens in XML, JSON etc.



Expose Business functionality over a network
web service is provides some sort of business functionality to client applicationsThis functionality can be invoked over the HTTP protocol which means that it can also be invoked over the internet. Nowadays all applications are on the internet which makes the purpose of Web services more useful.

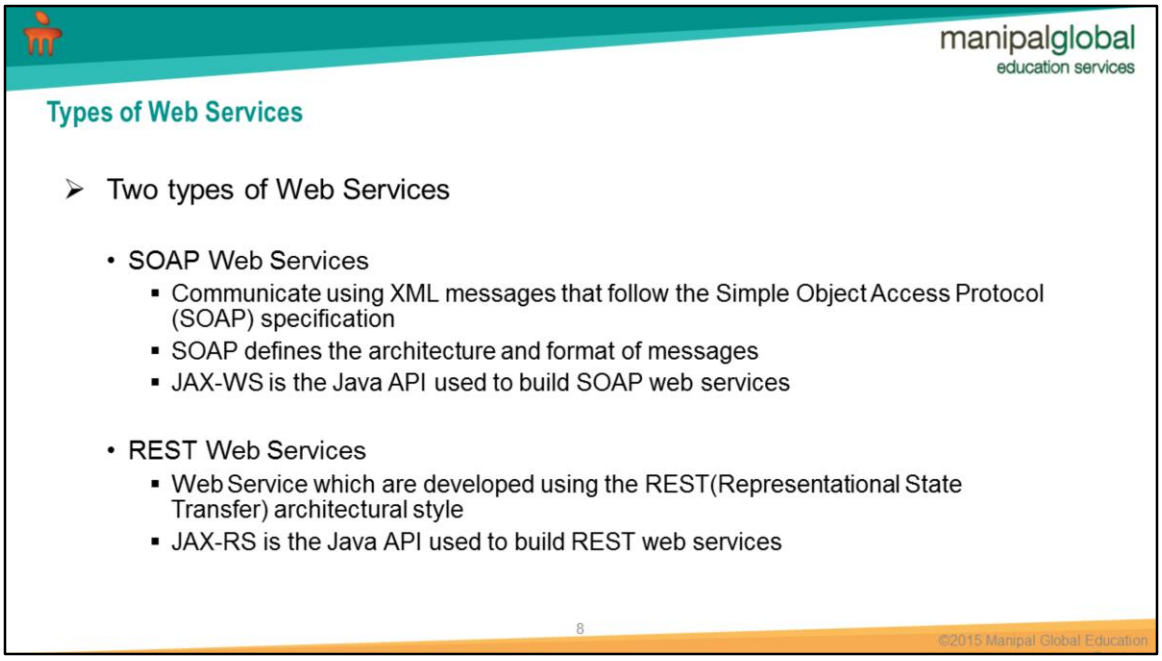
Interoperability amongst applications
Web services allow various applications to talk to each other and share data and services among themselves. Web Services typically work outside of private networks, offering developers a non-proprietary route to their solutions. Services developed are likely, therefore, to have a longer life-span, offering better return on investment of the developed service. Web Services also let developers use their preferred programming languages. In addition, thanks to the use of standards-based communications methods, Web Services are virtually platform-independent

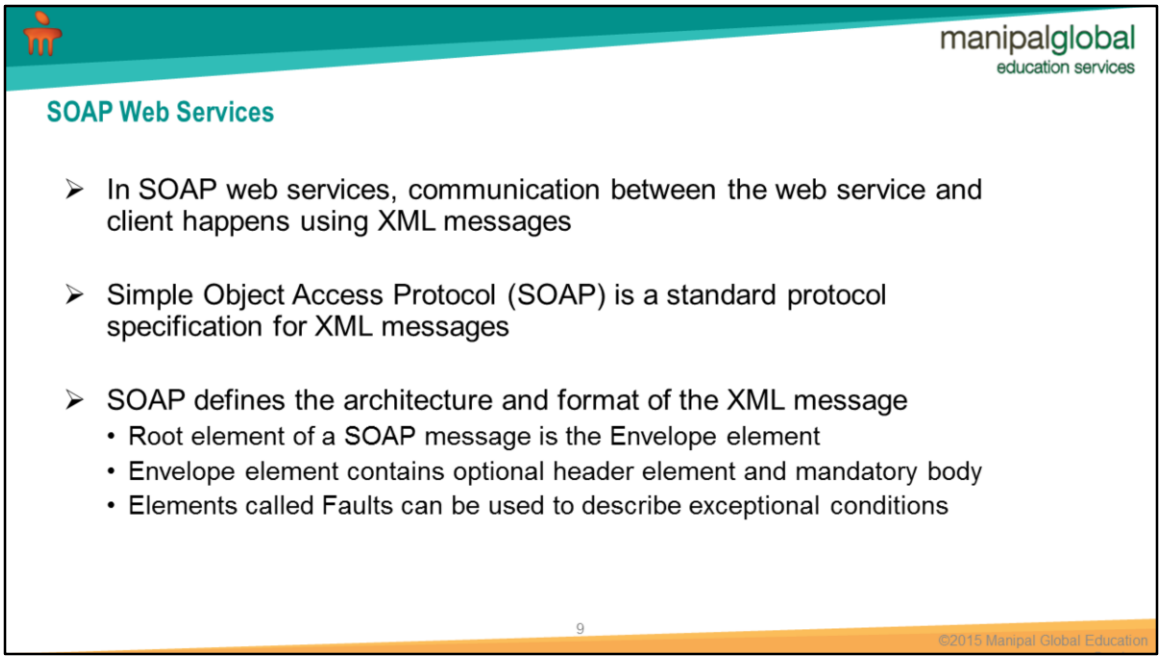
Standard protocol for exchanging messages
Web services use standardized industry protocol for the communication. Web services communicate over http. The HTTP standard allows more systems to communicate with one another. Exchanging of messages happens using SAOP

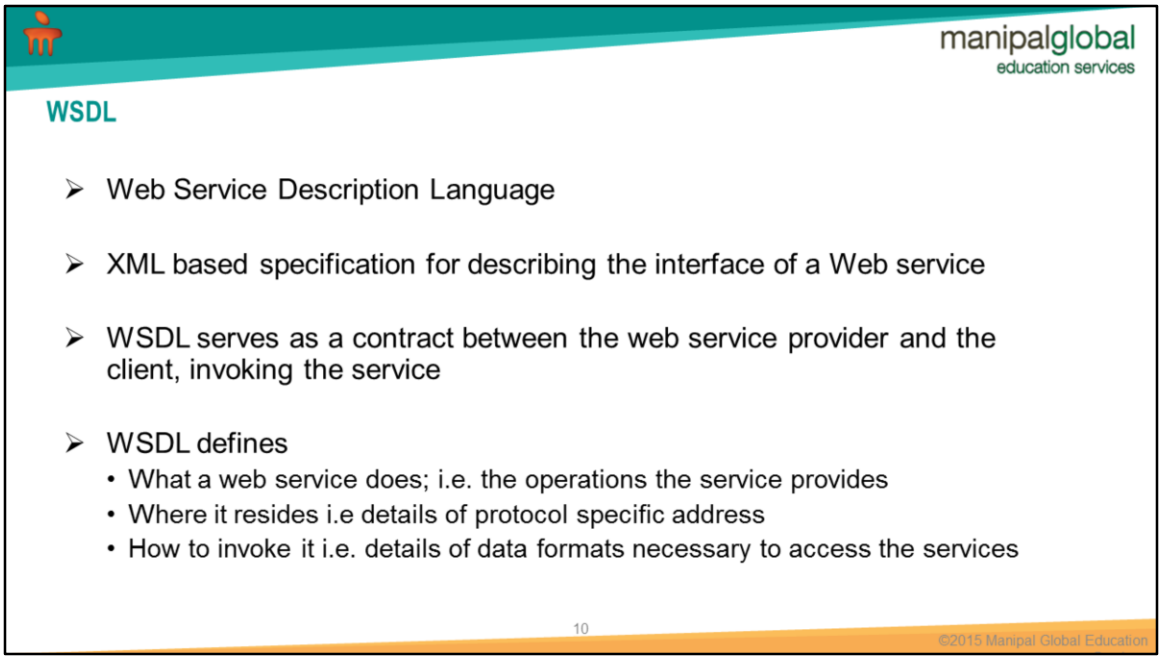
protocol for SOAP web services. SOAP is a XML based protocol which all platforms can easily access and process

Low Cost of Communication

Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.


manipalglobal
education services
Types of Web Services
Two types of Web Services
SOAP Web Services
Communicate using XML messages that follow the Simple Object Access Protocol (SOAP) specification
SOAP defines the architecture and format of messages
JAX-WS is the Java API used to build SOAP web services
REST Web Services
Web Service which are developed using the REST(Representational State Transfer) architectural style
JAX-RS is the Java API used to build REST web services
8
©2015 Manipal Global Education


manipalglobal
education services
SOAP Web Services
In SOAP web services, communication between the web service and client happens using XML messages
Simple Object Access Protocol (SOAP) is a standard protocol specification for XML messages
SOAP defines the architecture and format of the XML message
Root element of a SOAP message is the Envelope element
Envelope element contains optional header element and mandatory body
Elements called Faults can be used to describe exceptional conditions
9
©2015 Manipal Global Education


manipalglobal
education services
WSDL
Web Service Description Language
XML based specification for describing the interface of a Web service
WSDL serves as a contract between the web service provider and the client, invoking the service
WSDL defines
What a web service does; i.e. the operations the service provides
Where it resides i.e details of protocol specific address
How to invoke it i.e. details of data formats necessary to access the services
10
©2015 Manipal Global Education

UDDI

- Universal discovery and Description
- UDDI defines a scheme for publishing and finding web services of business entities
- Yellow pages of Web services

JAX-WS

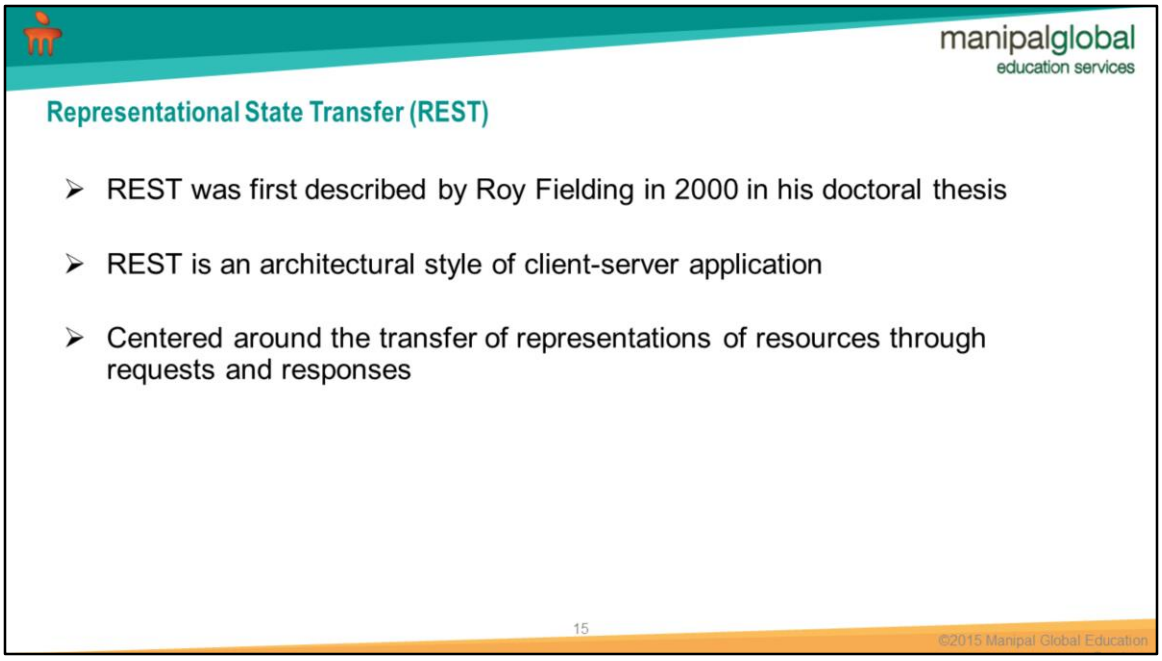
- Java API for XML based web services
- Web services created with JAX-WS are POJO's with annotations like
  - @WebService
  - @WebMethod
  - @WebParam etc

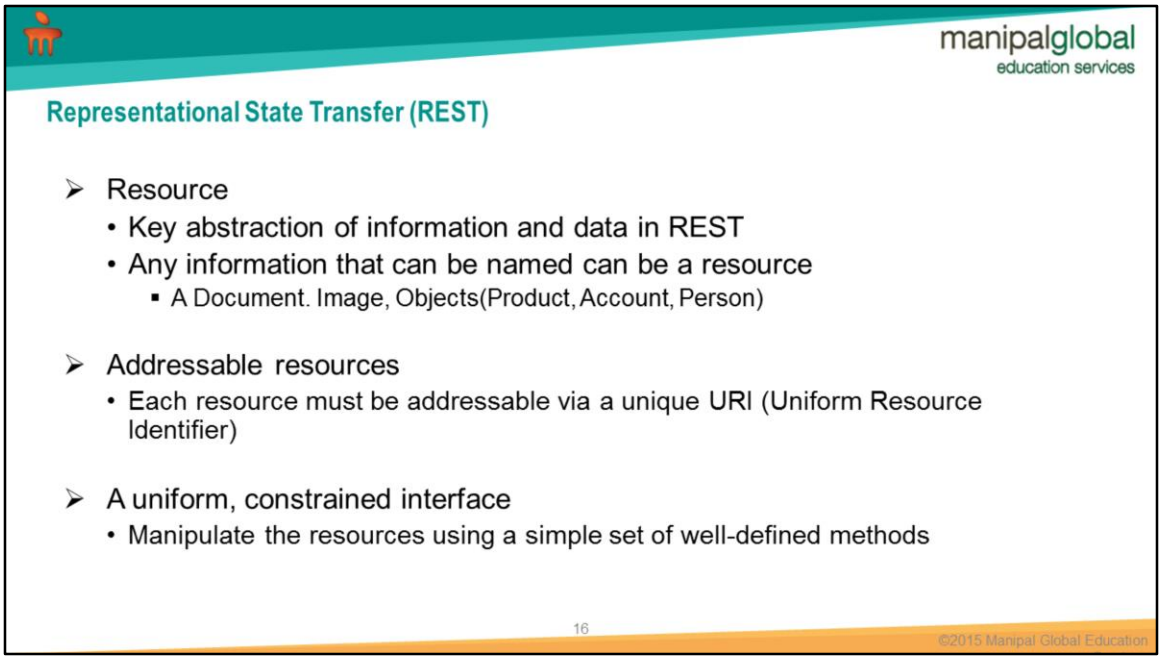
manipalglobal
education services

Simple SOAP web service

```
@WebService(name="MyCalculator")
public class myservice {

    @WebMethod(operationName="greet")
    public String sayHello(){
        return "Welcome to my Web Calculator";
    }

    @WebMethod(operationName="addNumbers")
    @WebResult(name="sum")
    public int add (int num1, int num2){
        return num1 + num2;
    }

    public static void main(String[] args) {
        Endpoint.publish("http://localhost:8181/myservice", new myservice());
    }
}
```

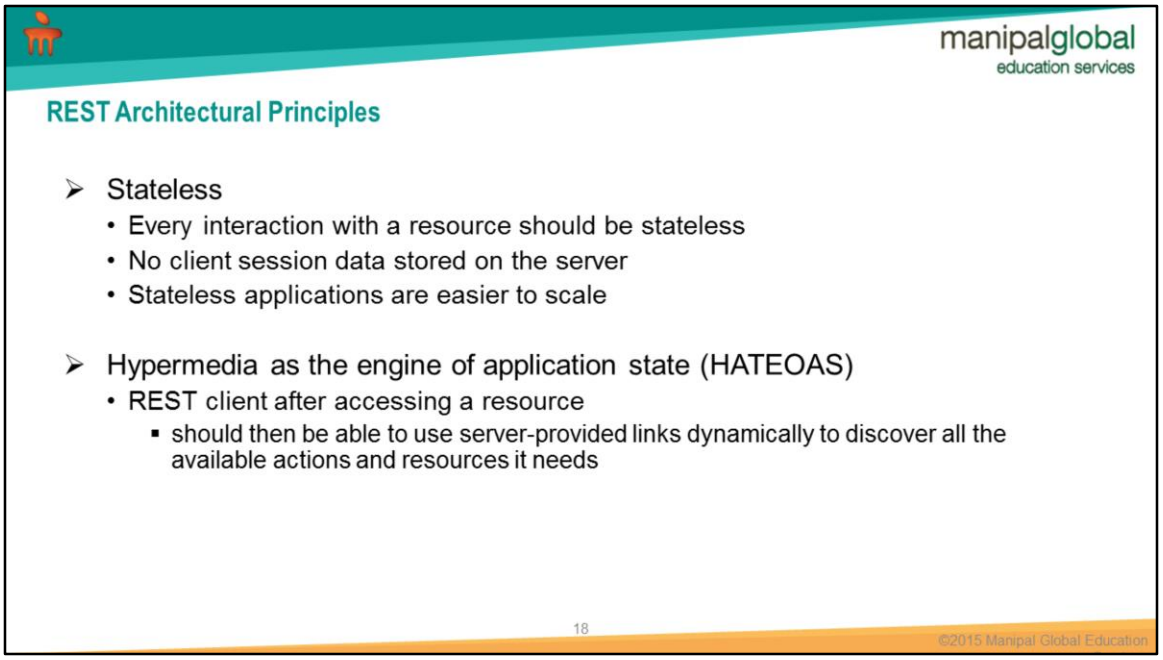

manipalglobal
education services
CONCEPT
REST Concepts
14
©2015 Manipal Global Education


manipalglobal
education services
Representational State Transfer (REST)
REST was first described by Roy Fielding in 2000 in his doctoral thesis
REST is an architectural style of client-server application
Centered around the transfer of representations of resources through requests and responses
15
©2015 Manipal Global Education


manipalglobal
education services
Representational State Transfer (REST)
Resource
Key abstraction of information and data in REST
Any information that can be named can be a resource
A Document. Image, Objects(Product, Account, Person)
Addressable resources
Each resource must be addressable via a unique URI (Uniform Resource Identifier)
A uniform, constrained interface
Manipulate the resources using a simple set of well-defined methods
16
©2015 Manipal Global Education

REST Architectural Principles

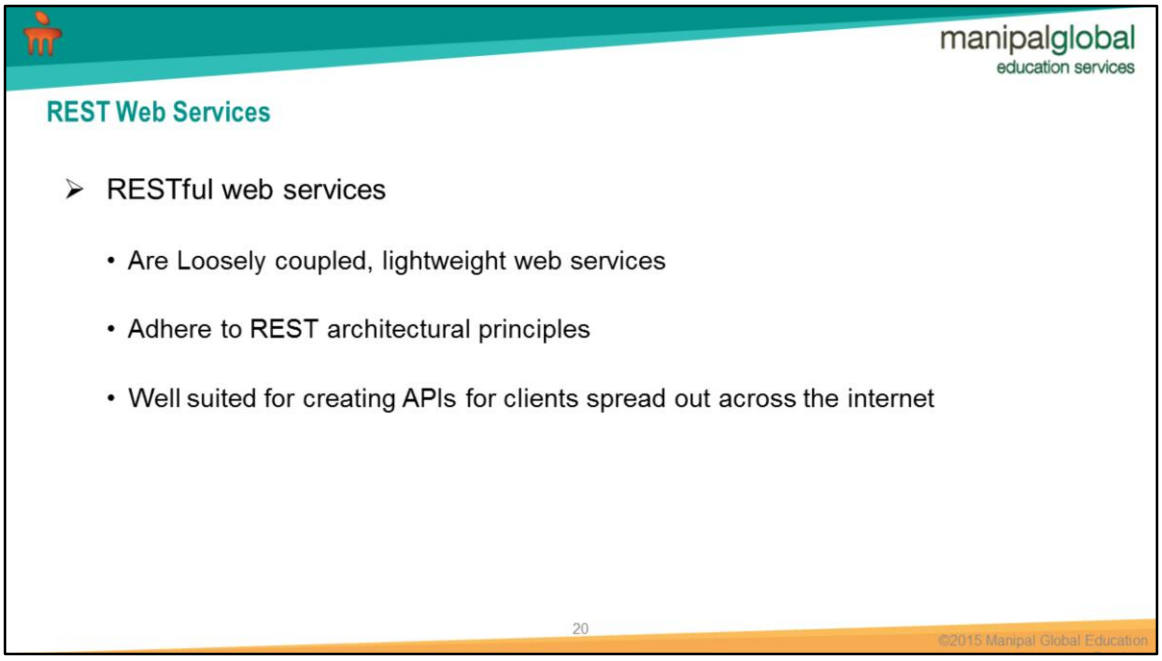
- Representation oriented
  - Clients and servers exchange representations of resources using a standardized protocol
  - State of a Resource can be represented many formats like XML, JSON, Plain text etc
  - Representation consists of data and metadata of the Resource
    - metadata includes representation format, error details, authentication details
  - REST components perform actions on a resource by using a representation

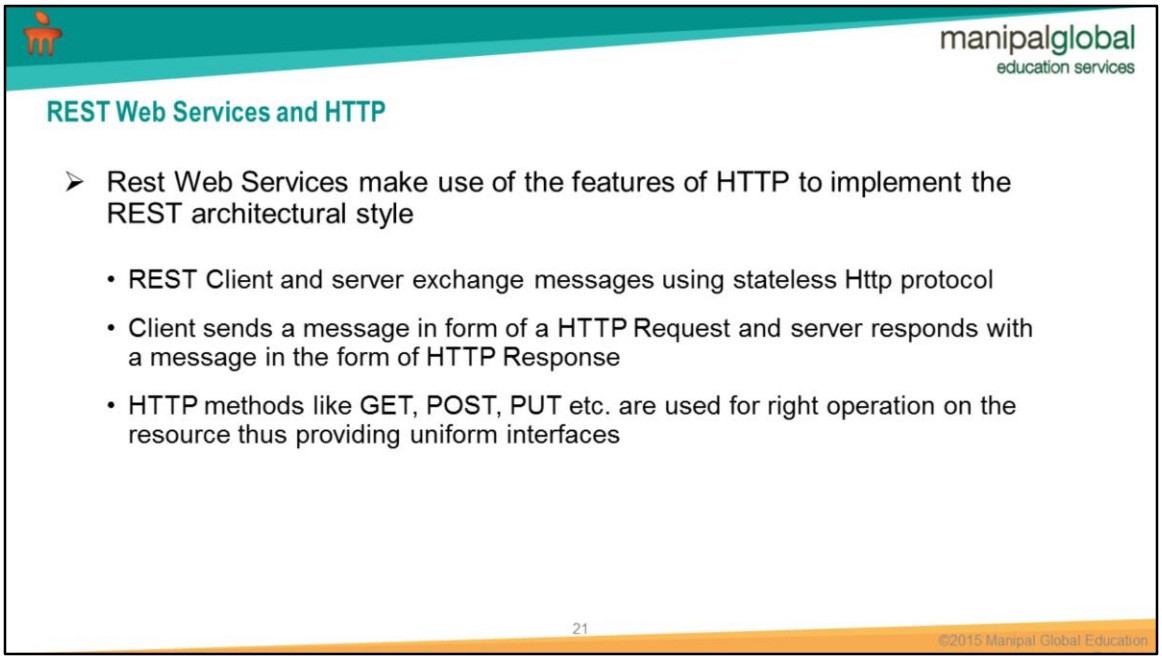


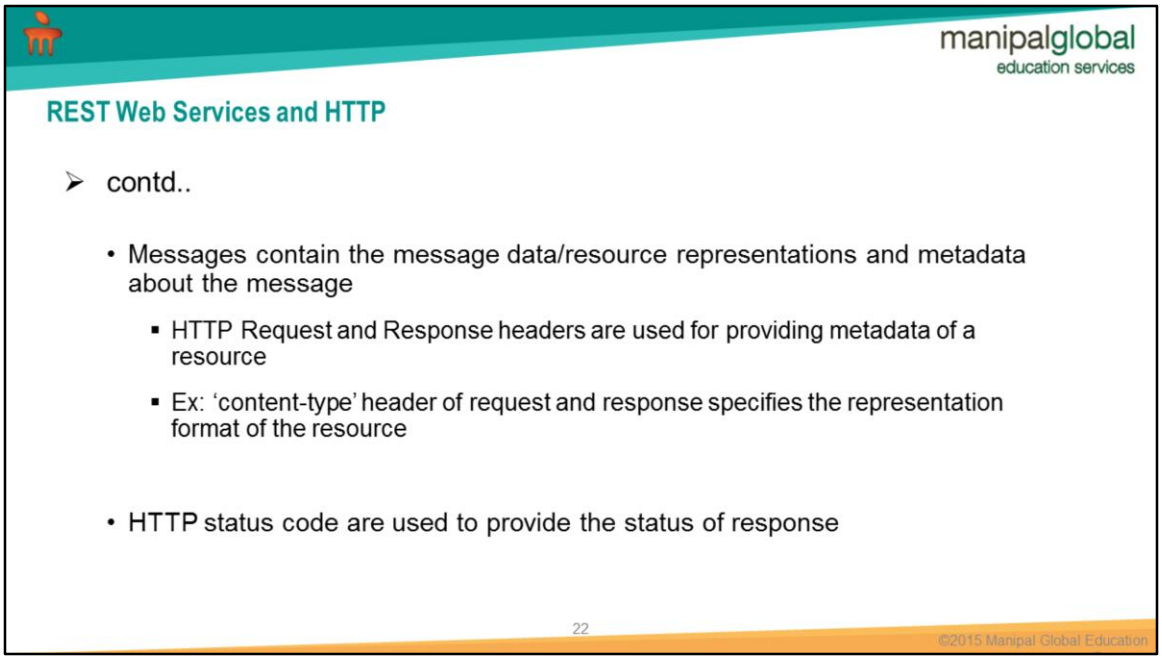
Stateless

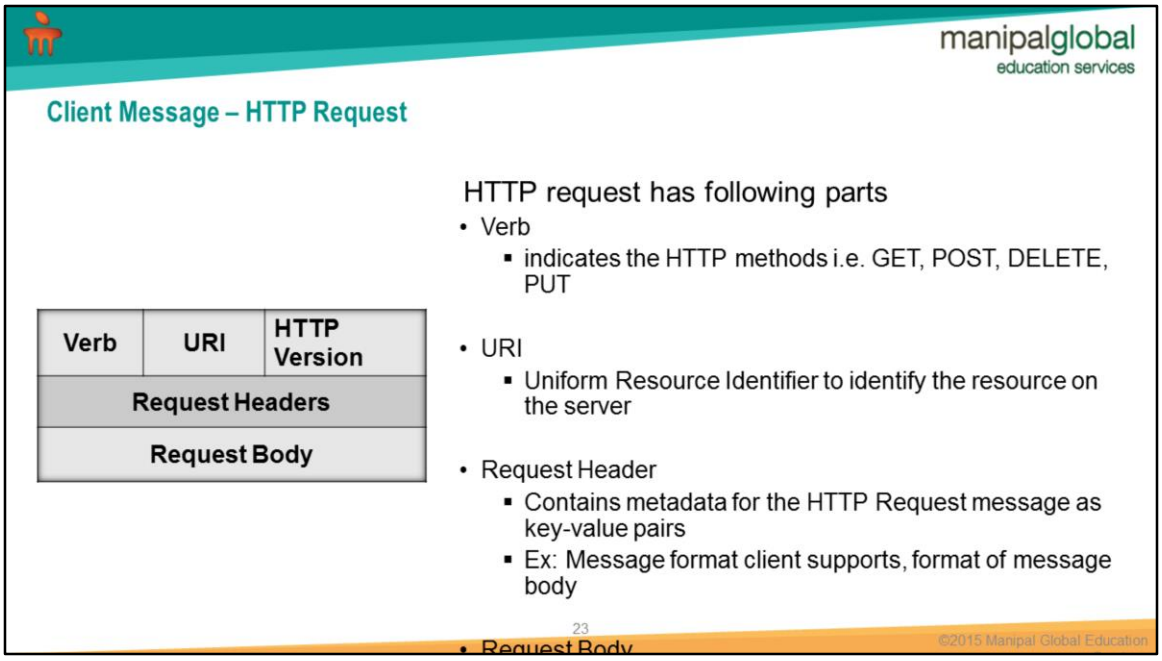
The client–server communication is constrained by no client context being stored on the server between requests. Each request from any client contains all the information necessary to service the request, and session state is held in the client. The session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication. The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be *in transition*. The representation of each application state contains links that may be used the next time the client chooses to initiate a new state-transition.

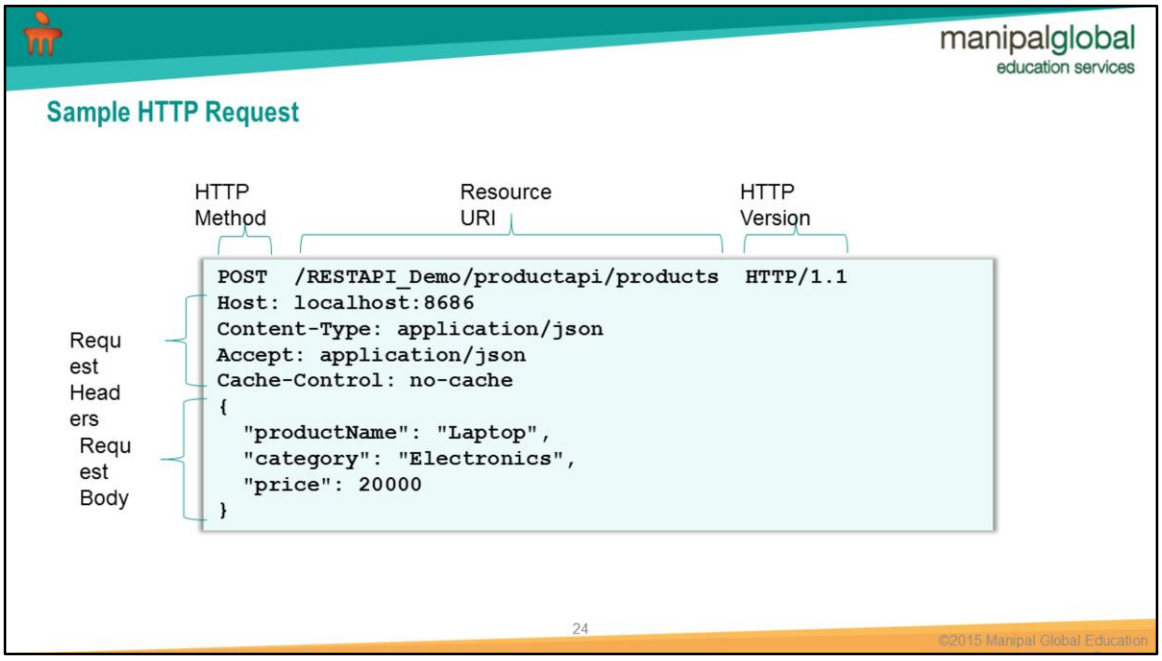

manipalglobal
education services
CONCEPT
REST Web Services
19
©2015 Manipal Global Education

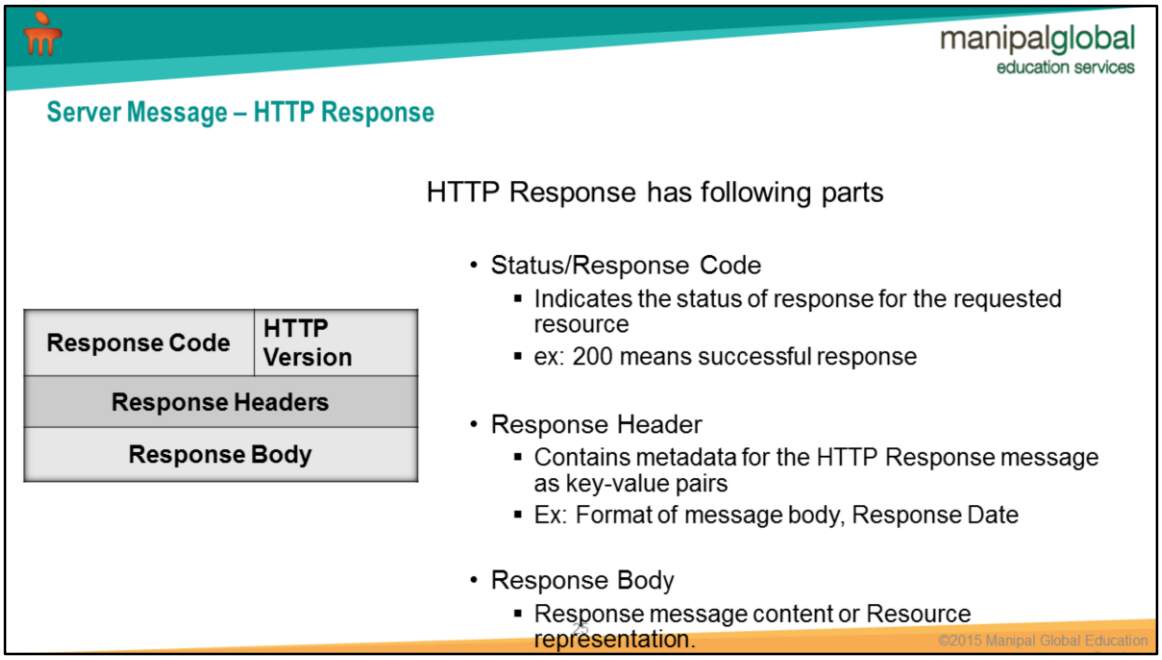

manipalglobal
education services
REST Web Services
RESTful web services
Are Loosely coupled, lightweight web services
Adhere to REST architectural principles
Well suited for creating APIs for clients spread out across the internet
20
©2015 Manipal Global Education

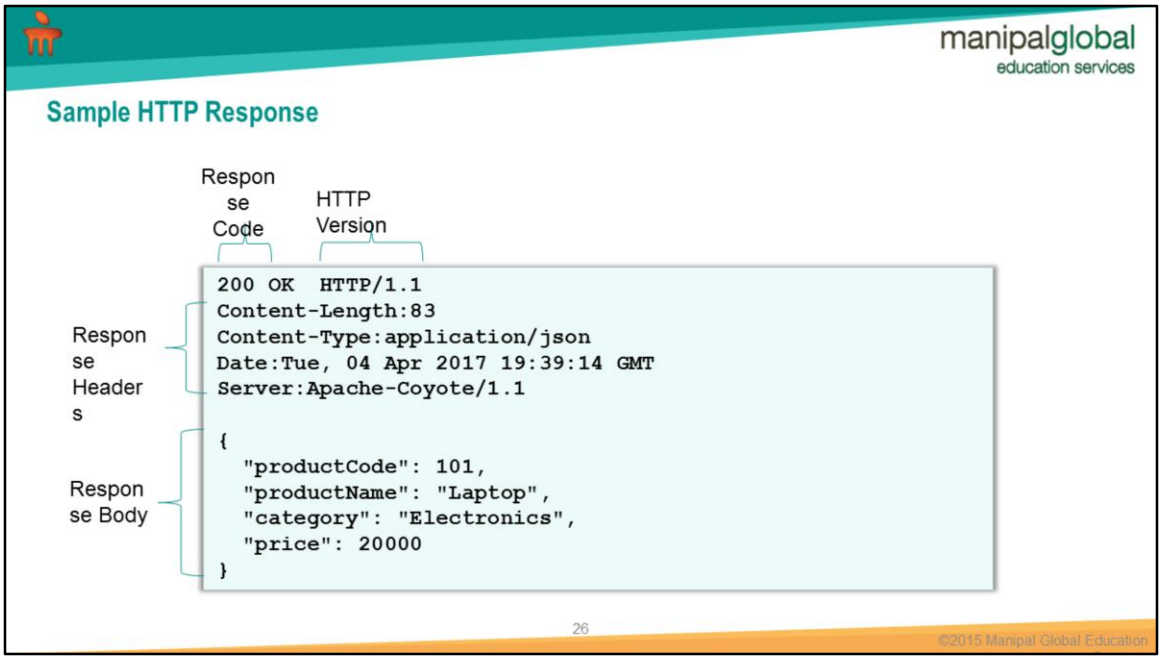

manipalglobal
education services
REST Web Services and HTTP
Rest Web Services make use of the features of HTTP to implement the REST architectural style
REST Client and server exchange messages using stateless Http protocol
Client sends a message in form of a HTTP Request and server responds with a message in the form of HTTP Response
HTTP methods like GET, POST, PUT etc. are used for right operation on the resource thus providing uniform interfaces
21
©2015 Manipal Global Education

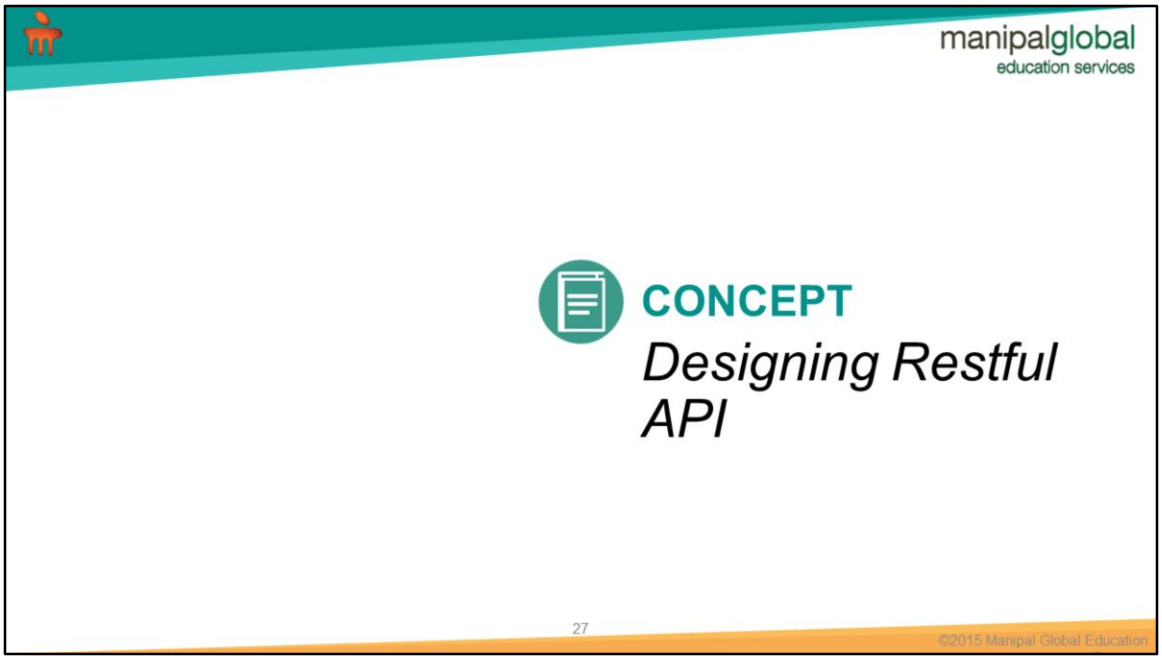

manipalglobal
education services
REST Web Services and HTTP
contd..
Messages contain the message data/resource representations and metadata about the message
HTTP Request and Response headers are used for providing metadata of a resource
Ex: 'content-type' header of request and response specifies the representation format of the resource
HTTP status code are used to provide the status of response
22
©2015 Manipal Global Education


manipalglobal
education services
Client Message – HTTP Request
HTTP request has following parts
• Verb
▪ indicates the HTTP methods i.e. GET, POST, DELETE, PUT
Verb
URI
HTTP Version
Request Headers
Request Body
• URI
▪ Uniform Resource Identifier to identify the resource on the server
• Request Header
▪ Contains metadata for the HTTP Request message as key-value pairs
▪ Ex: Message format client supports, format of message body
23
• Request Body
©2015 Manipal Global Education


manipalglobal
education services
Sample HTTP Request
HTTP Method
Resource URI
HTTP Version
POST /RESTAPI_Demo/productapi/products HTTP/1.1
Host: localhost:8686
Content-Type: application/json
Accept: application/json
Cache-Control: no-cache
{
"productName": "Laptop",
"category": "Electronics",
"price": 20000
}
Request Headers
Request Body
24
©2015 Manipal Global Education

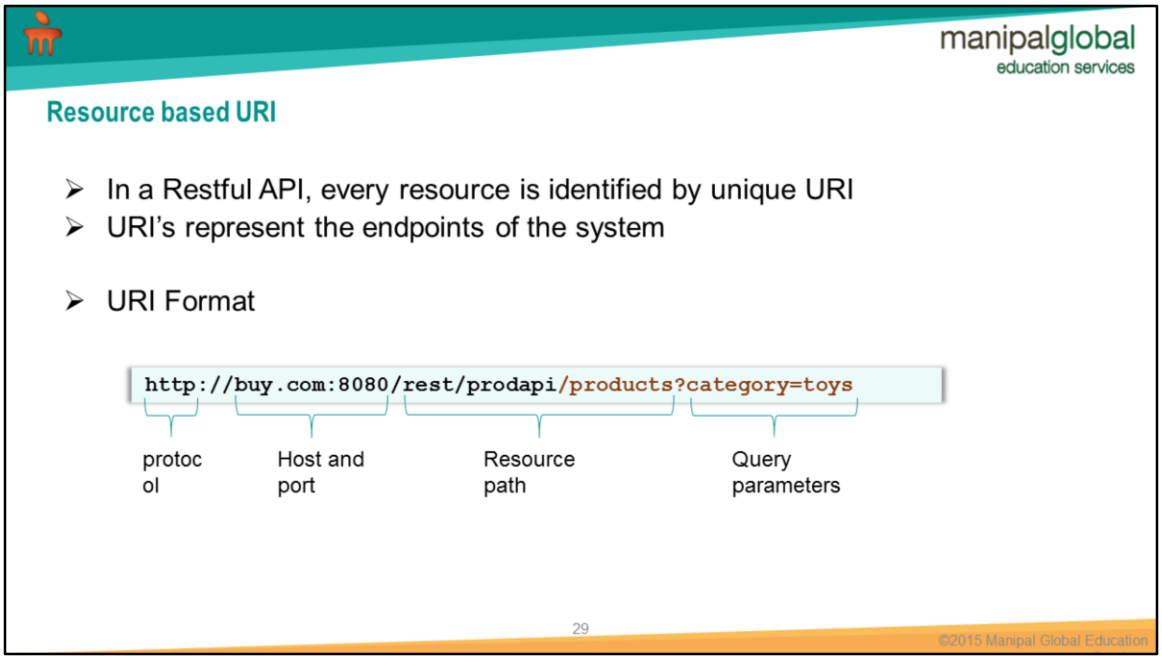

manipalglobal
education services
Server Message – HTTP Response
HTTP Response has following parts
Response Code
HTTP Version
Response Headers
Response Body
• Status/Response Code
▪ Indicates the status of response for the requested resource
▪ ex: 200 means successful response
• Response Header
▪ Contains metadata for the HTTP Response message as key-value pairs
▪ Ex: Format of message body, Response Date
• Response Body
▪ Response message content or Resource representation.
©2015 Manipal Global Education

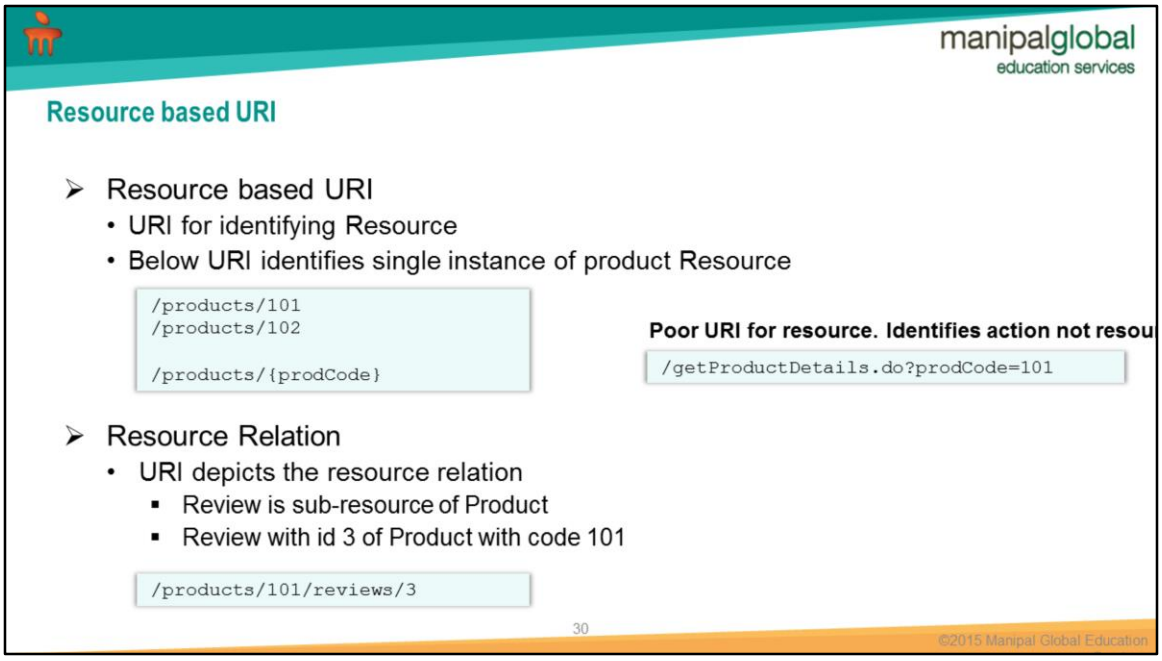

manipalglobal
education services
Sample HTTP Response
Respon se Code
HTTP Version
200 OK HTTP/1.1
Content-Length:83
Content-Type:application/json
Date:Tue, 04 Apr 2017 19:39:14 GMT
Server:Apache-Coyote/1.1
{
"productCode": 101,
"productName": "Laptop",
"category": "Electronics",
"price": 20000
}
Respon se Header s
Respon se Body
26
©2015 Manipal Global Education

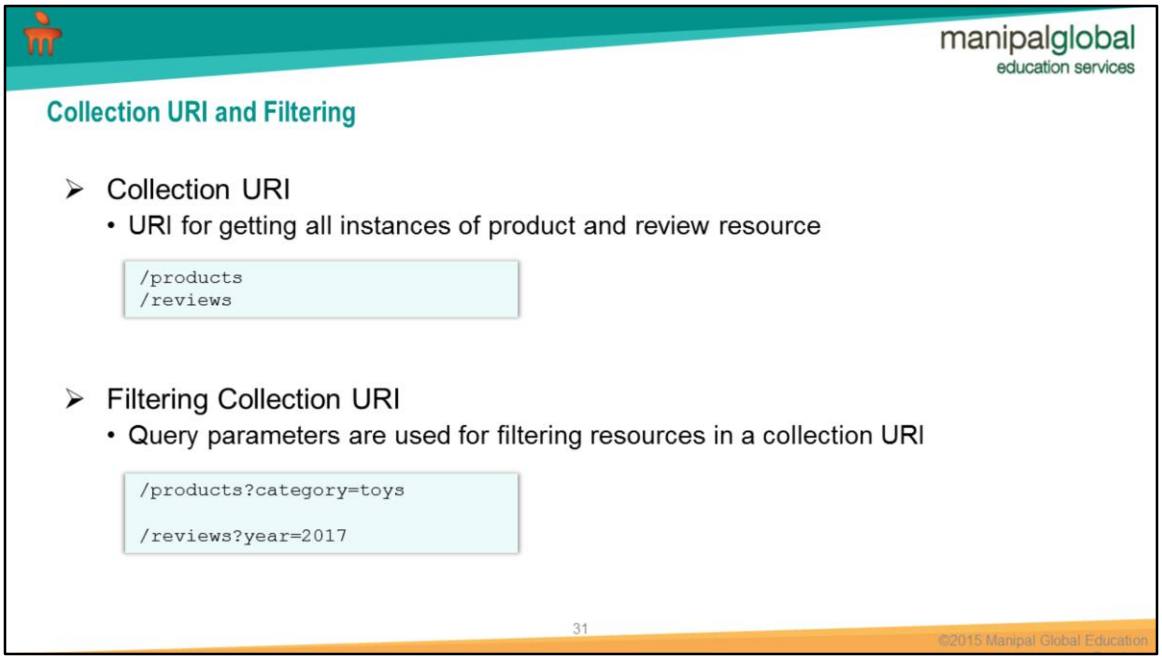

manipalglobal
education services
CONCEPT
Designing Restful API
27
©2015 Manipal Global Education

Designing Restful API

- Lets consider a simple application which maintains Product Information and customer reviews for products
- The Apllication needs to provide the following web services to other applications
  - Add and Delete products
  - Get product details for one product and all products
  - Modify product details
  - Add and Get Reviews
- Noun/things in a system helps identifying possible resources
  - In above case, Product and Review can be identified as Resources


manipalglobal
education services
Resource based URI
In a Restful API, every resource is identified by unique URI
URI's represent the endpoints of the system
URI Format
http://buy.com:8080/rest/prodapi/products?category=toys
protoc ol
Host and port
Resource path
Query parameters
29
©2015 Manipal Global Education


manipalglobal
education services
Resource based URI
Resource based URI
URI for identifying Resource
Below URI identifies single instance of product Resource
/products/101
/products/102
/products/{prodCode}
Poor URI for resource. Identifies action not resou
/getProductDetails.do?prodCode=101
Resource Relation
URI depicts the resource relation
Review is sub-resource of Product
Review with id 3 of Product with code 101
/products/101/reviews/3
30
©2015 Manipal Global Education


manipalglobal
education services
Collection URI and Filtering
Collection URI
URI for getting all instances of product and review resource
/products
/reviews
Filtering Collection URI
Query parameters are used for filtering resources in a collection URI
/products?category=toys
/reviews?year=2017
31
©2015 Manipal Global Education

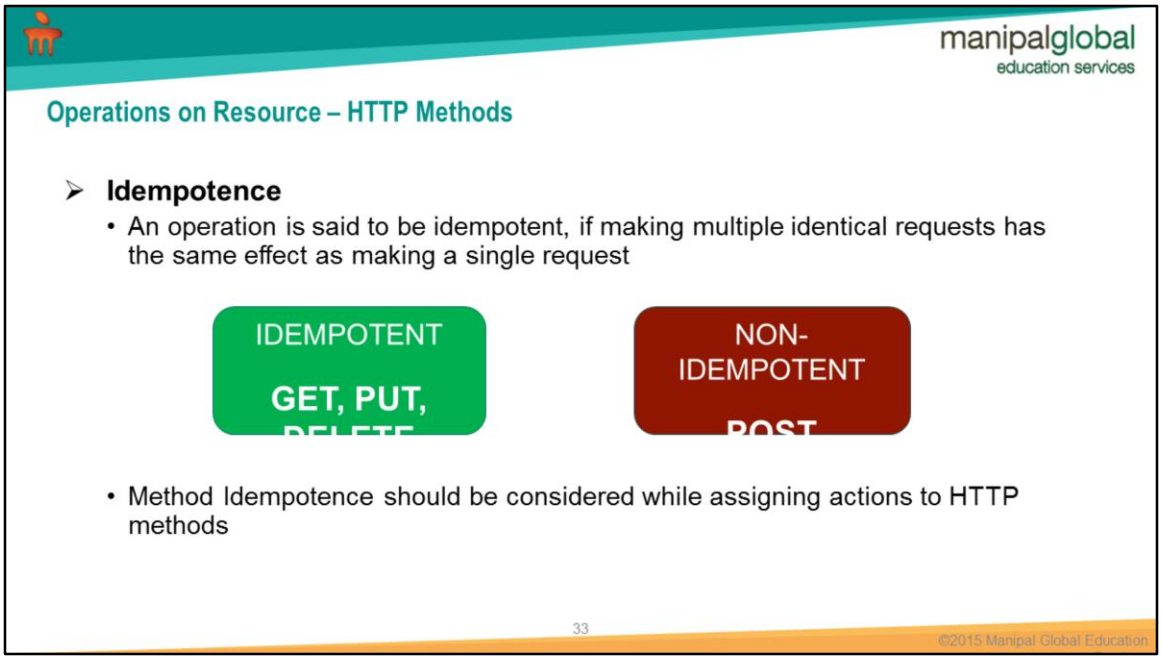
manipalglobal
education services

Operations on Resource – HTTP Methods

- Resource URI by itself does not specify the action to be taken on the resource
- REST Web service uses HTTP method to specify the action/operation

- Right HTTP method should be chosen for different actions / operations
  - GET - Used for read operation for a resource
  - POST - Used for inserting a new resource
  - PUT - Used for updating a resource
  - DELETE - Used for deleting a resource

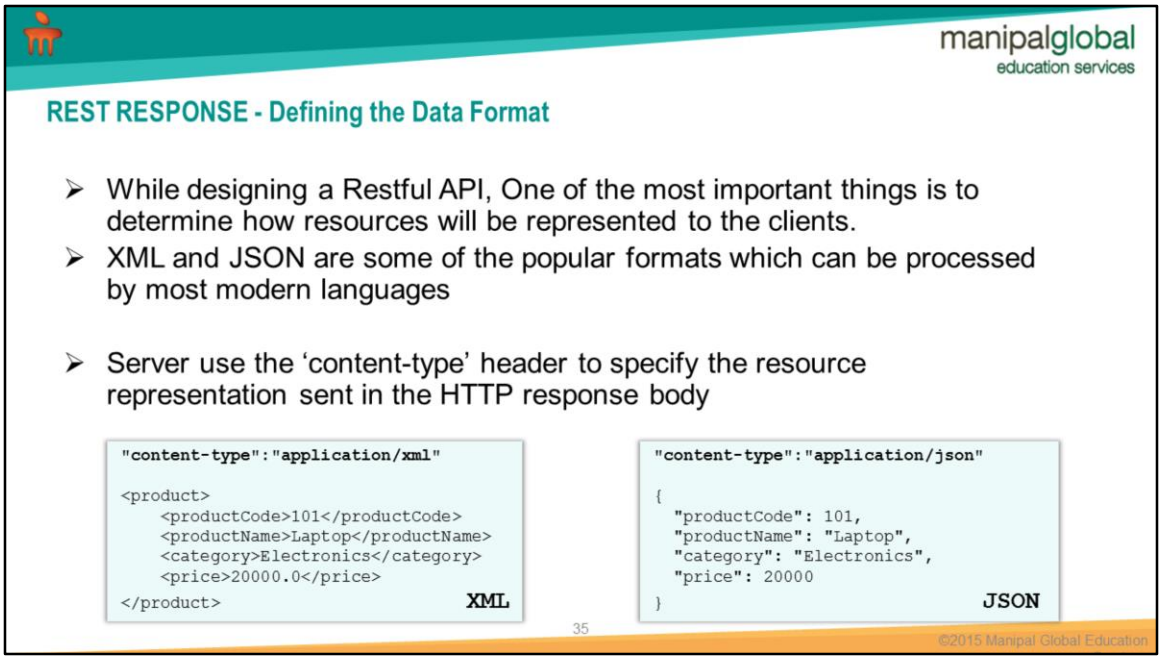
- HEAD and options method are also supported by REST Web services


manipalglobal
education services
Operations on Resource – HTTP Methods
Idempotence
An operation is said to be idempotent, if making multiple identical requests has the same effect as making a single request
IDEMPOTENT
GET, PUT,
NON-
IDEMPOTENT
POST
Method Idempotence should be considered while assigning actions to HTTP methods
33
©2015 Manipal Global Education

manipalglobal
education services

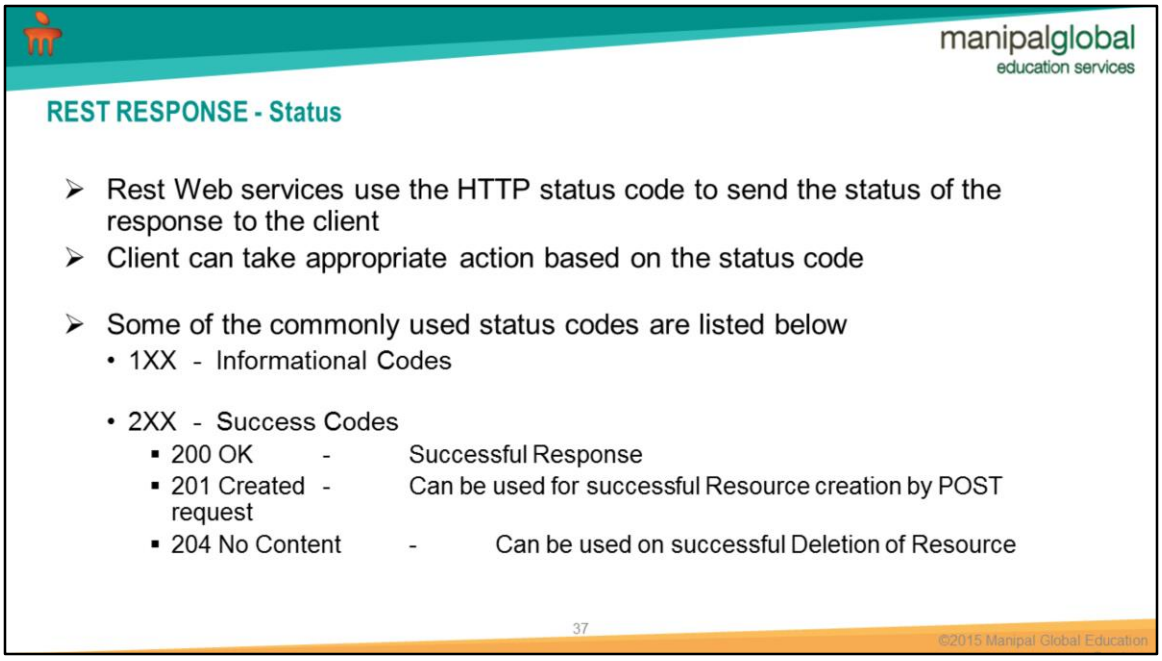
Assigning HTTP methods to URI's

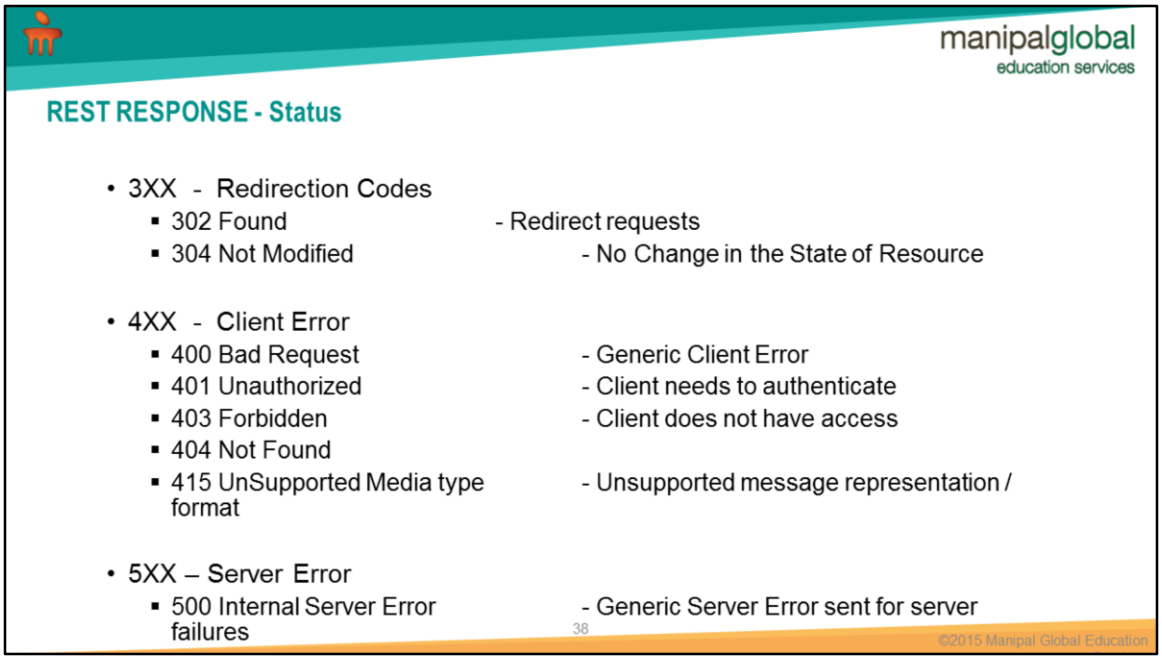
HTTP Method	URI	Operation
GET	/products/101	Gets product details for product code 101
DELETE	/products/101	Delete product with product code 101
PUT	/products/101	Update product with product code 101
POST	/products	Add new product. (typically POST operation do not take a ID, since ID will be assigned by the application
GET	/products	Get product details for all products
GET	/products?category=toys	Get product details for all products belonging to category toys
POST	/products/101/reviews	Add new review for product with product code 101


manipalglobal
education services
REST RESPONSE - Defining the Data Format
While designing a Restful API, One of the most important things is to determine how resources will be represented to the clients.
XML and JSON are some of the popular formats which can be processed by most modern languages
Server use the 'content-type' header to specify the resource representation sent in the HTTP response body
"content-type":"application/xml"
<product>
<productCode>101</productCode>
<productName>Laptop</productName>
<category>Electronics</category>
<price>20000.0</price>
</product>
XML
"content-type":"application/json"
{
"productCode": 101,
"productName": "Laptop",
"category": "Electronics",
"price": 20000
}
JSON
35
©2015 Manipal Global Education

REST RESPONSE - Content Negotiation

- The Server application can define API for representing the same resource in multiple representations
- Depending on the resource representation the clients ask for, Server sends the requested representation, if supported
  - If client accepts XML, Server can send the XML representation of the resource.
  - If client accepts JSON, Server can send the JSON representation of the same resource
- Client and Server negotiate the content type for the resource using the headers in request


manipalglobal
education services
REST RESPONSE - Status
Rest Web services use the HTTP status code to send the status of the response to the client
Client can take appropriate action based on the status code
Some of the commonly used status codes are listed below
1XX - Informational Codes
2XX - Success Codes
200 OK - Successful Response
201 Created - Can be used for successful Resource creation by POST request
204 No Content - Can be used on successful Deletion of Resource
37
©2015 Manipal Global Education


manipalglobal
education services
REST RESPONSE - Status
• 3XX - Redirection Codes
▪ 302 Found - Redirect requests
▪ 304 Not Modified - No Change in the State of Resource
• 4XX - Client Error
▪ 400 Bad Request - Generic Client Error
▪ 401 Unauthorized - Client needs to authenticate
▪ 403 Forbidden - Client does not have access
▪ 404 Not Found
▪ 415 UnSupported Media type format - Unsupported message representation /
• 5XX – Server Error
▪ 500 Internal Server Error failures - Generic Server Error sent for server
38
©2015 Manipal Global Education


manipalglobal
education services
CONCEPT
Building Rest Web Services using JAX-RS
39
©2015 Manipal Global Education


manipalglobal
education services
JAX-RS
Acronym for "Java API for RESTFul Web Services"
JAX-RS
Java API specification defined to support the development of REST Web Services
Simplifies the development and deployment of REST web service endpoints and clients
Following are some of the implementations of JAX-RS API
Jersey
Apache CXF
RESTeasy
Restlet
Application Server's like WebSphere, WebLogic
Jersey is the reference implementation of JAX-RS from Oracle
40
©2015 Manipal Global Education

manipalglobal
education services

JAX-RS Features

- JAX-RS API provides annotations to simplify the development of RESTful web services
- Annotations are applied on plain java classes and methods. They help to
  - Bind a URI pattern to a resource
  - Bind HTTP operations to individual methods of a resource
  - Extract values of query and form parameters, headers, cookies, and other pieces of the HTTP request message in to Java parameters and fields
  - Identify the available content types that can be consumed or produced by a resource

JAX-RS Features

- JAX-RS has Message body readers and writers
  - Allows to decouple data format marshalling and unmarshalling from Java data objects
- JAX-RS has Exception mappers
  - Maps application-thrown exception to an HTTP response code and message
- JAX-RS annotation are defined in package

```
javax.ws.rs
```

Setting Up JAX-RS project using Jersey in Eclipse – Using Maven

- Configure Apache Tomcat Server in Eclipse
- Create a Maven project in Eclipse with archetype maven-archetype-webapp
- Add following dependency in pom.xml

```
<dependency>
	<groupId>org.glassfish.jersey.containers</groupId>
	<artifactId>jersey-container-servlet</artifactId>
	<version>2.25.1</version>
</dependency>
<!-- For JSON support add the following-->
<dependency>
	<groupId>org.glassfish.jersey.media</groupId>
	<artifactId>jersey-media-moxy</artifactId>
	<version>2.25.1</version>
</dependency>
```

manipalglobal
education services

Setting Up JAX-RS project using Jersey in Eclipse

- Download the latest version of Jersey distribution as zip from https://jersey.java.net/download.html.
- The zip contains the Jersey implementation JAR and its core dependencies. Extract the Jersey bundle zip file. Jersey jars are present in folders api, lib, ext of the extracted zip file
- Configure Apache Tomcat Server in Eclipse.
- Create a Dynamic web project in Eclipse
- Add the jersey jars in the build path and deployment assembly.

manipalglobal
education services

Registering JAX-RS Resources

- A JAX-RS application consists of one or more Resource/Provider classes
- Resources/Providers are java classes annotated with JAX-RS annotations
- These Resources/Providers have to be registered with JAX-RS runtime

- Two ways to configure Resources and Provides with JAX-RS
  - Create a subclass of javax.ws.rs.core.Application annotated with @ApplicationPath
  - Configure using jersey Servlet in web.xml

manipalglobal
education services

Registering JAX-RS Resources using Application Subclass

- JAX-RS provides a class javax.ws.rs.core.Application for registering Resources and Providers
  - Create a subclass of Application
  - Add class level @ApplicationPath annotation to set the base URI

```
@ApplicationPath("productapi")
public class ProductApp extends Application{

}
```

- All resources defined within the application are relative to base URI
- By default, Jersey Runtime will scan all the classes in the build path for registering resources

manipalglobal
education services

Registering JAX-RS Resources

- To manually register Resource classes in the Application, Override the getClasses() method

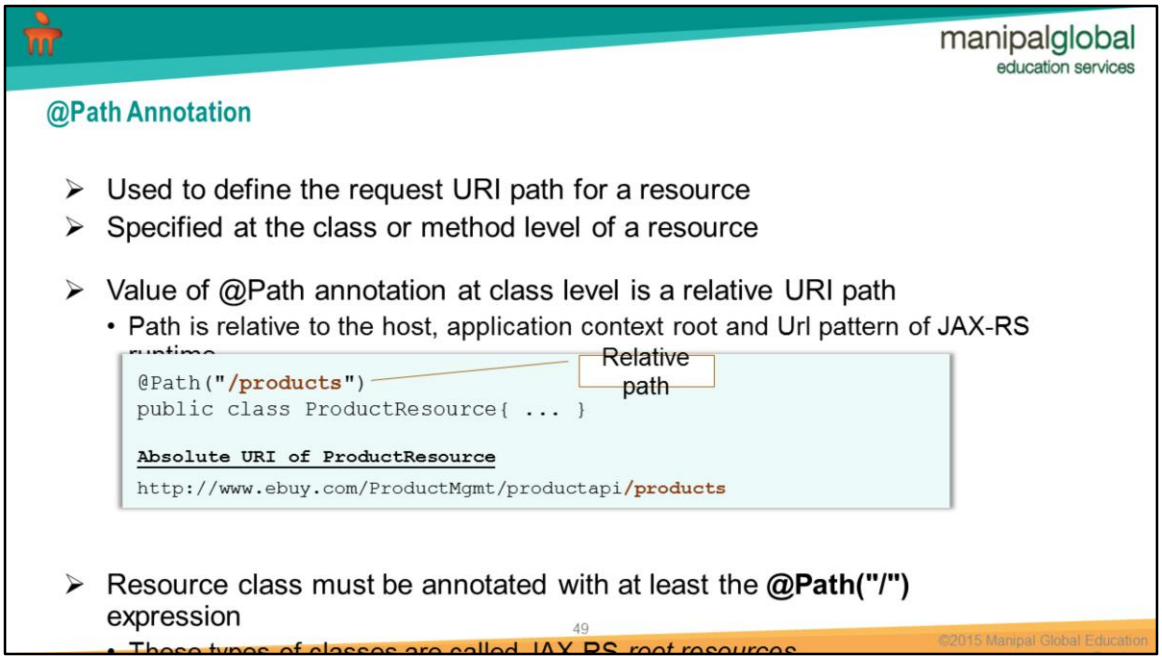
```
@ApplicationPath("productapi")
public class ProductApp extends Application{

    @Override
    public Set<Class<?>> getClasses() {
        final Set<Class<?>> classes = new HashSet<>();
        // register root resource
        classes.add(MyResource.class);
        classes.add(ProductResource.class);
        return classes;
    }
}
```

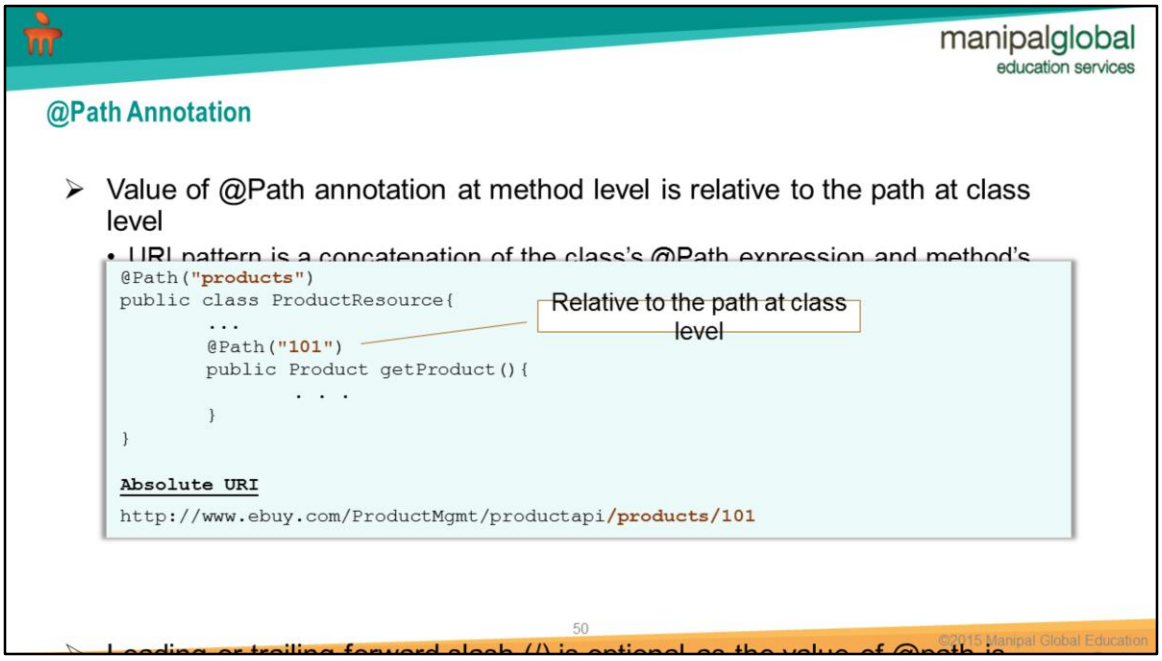
- In the above example, Only two classes, MyResource and ProductResource will be registered with JAX-RS runtime

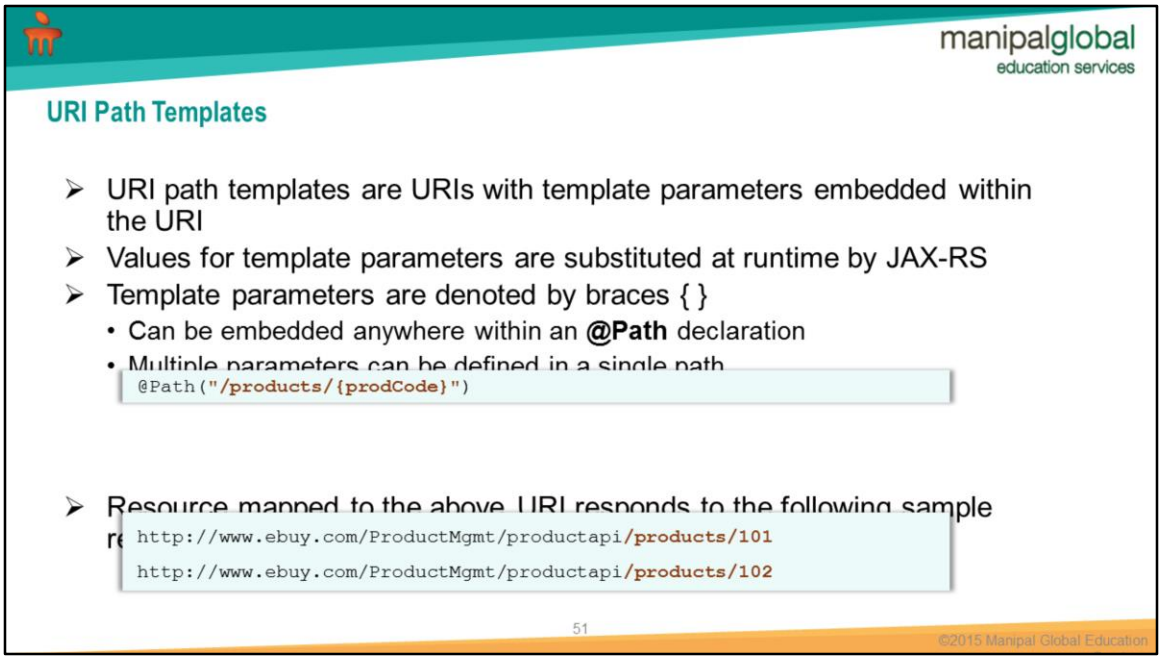
manipalglobal
education services

Registering JAX-RS Resources using web.xml

```
<servlet>
   <servlet-name>Jersey Servlet</servlet-name>
   <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
   <init-param>
       <param-name>jersey.config.server.provider.packages</param-name>
       <param-value>com.mpl.resources</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
   <servlet-name>Jersey Servlet</servlet-name>
    <url-pattern>/productapi/*</url-pattern>
</servlet-mapping>
```


manipalglobal
education services
@Path Annotation
Used to define the request URI path for a resource
Specified at the class or method level of a resource
Value of @Path annotation at class level is a relative URI path
Path is relative to the host, application context root and Url pattern of JAX-RS
@Path("/products")
public class ProductResource{ ... }
Relative path
Absolute URI of ProductResource
http://www.ebuy.com/ProductMgmt/productapi/products
Resource class must be annotated with at least the @Path("/") expression
49
©2015 Manipal Global Education


manipalglobal
education services
@Path Annotation
Value of @Path annotation at method level is relative to the path at class level
URI pattern is a concatenation of the class's @Path expression and method's
@Path("products")
public class ProductResource{
...
@Path("101")
public Product getProduct(){
. . .
}
}
Relative to the path at class level
Absolute URI
http://www.ebuy.com/ProductMgmt/productapi/products/101
50
©2015 Manipal Global Education
Leading or trailing forward slash (/) is optional as the value of @path is


manipalglobal
education services
URI Path Templates
URI path templates are URIs with template parameters embedded within the URI
Values for template parameters are substituted at runtime by JAX-RS
Template parameters are denoted by braces { }
Can be embedded anywhere within an @Path declaration
Multiple parameters can be defined in a single path
@Path("/products/{prodCode}")
Resource mapped to the above URI responds to the following sample
http://www.ebuy.com/ProductMgmt/productapi/products/101
http://www.ebuy.com/ProductMgmt/productapi/products/102
51
©2015 Manipal Global Education

manipalglobal
education services

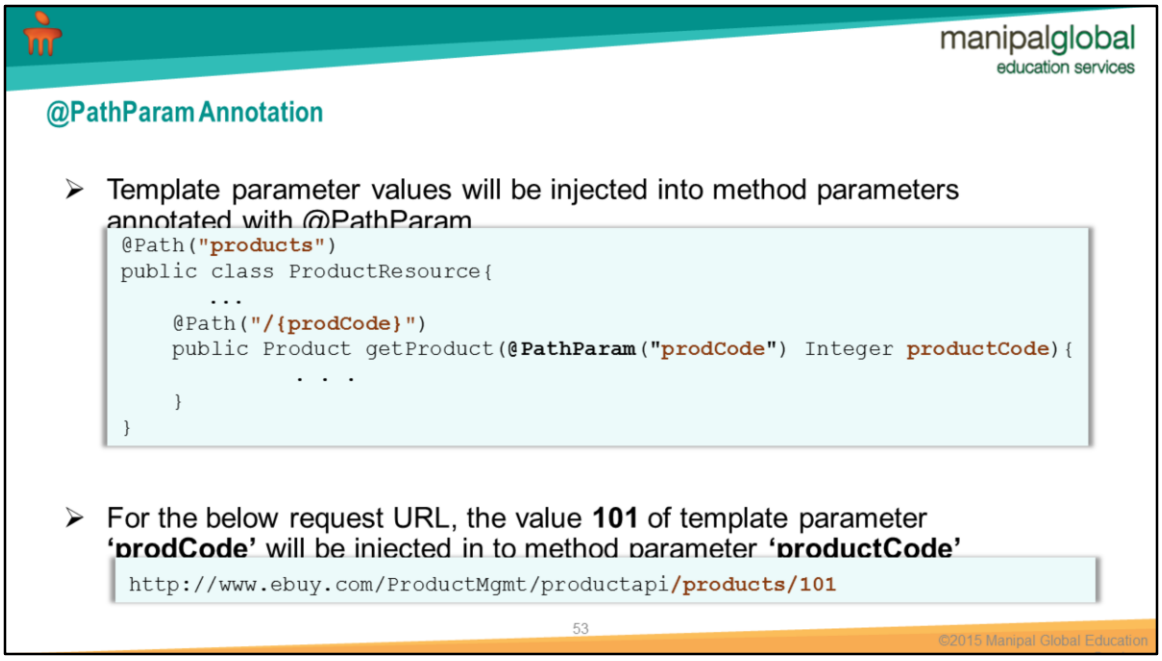
Multiple template Parameters

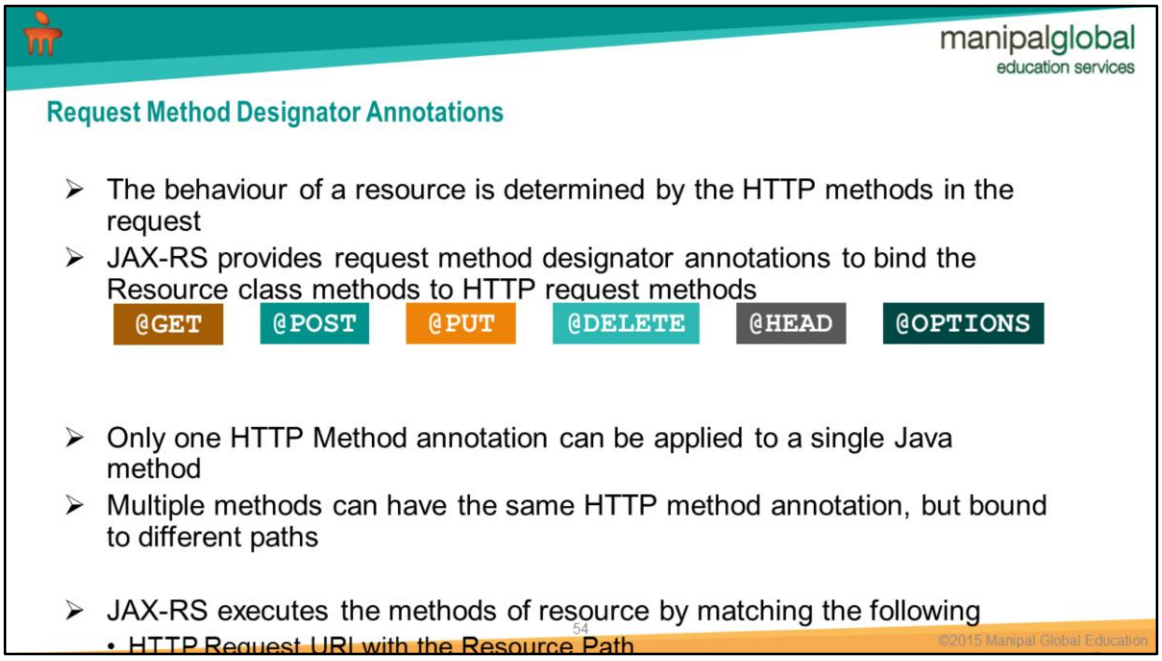
- URI path templates with multiple template parameters

```
@Path("/products/{prodCode}/reviews/{reviewId}")
```

- Resource mapped to the above URI responds to the following sample request URL's

```
http://www.ebuy.com/ProductMgmt/productapi/products/101/reviews/1
http://www.ebuy.com/ProductMgmt/productapi/products/102/reviews/1
```


manipalglobal
education services
@PathParam Annotation
Template parameter values will be injected into method parameters annotated with @PathParam
@Path("products")
public class ProductResource{
...
@Path("/{prodCode}")
public Product getProduct(@PathParam("prodCode") Integer productCode){
. . .
}
}
For the below request URL, the value 101 of template parameter 'prodCode' will be injected in to method parameter 'productCode'
http://www.ebuy.com/ProductMgmt/productapi/products/101
53
©2015 Manipal Global Education

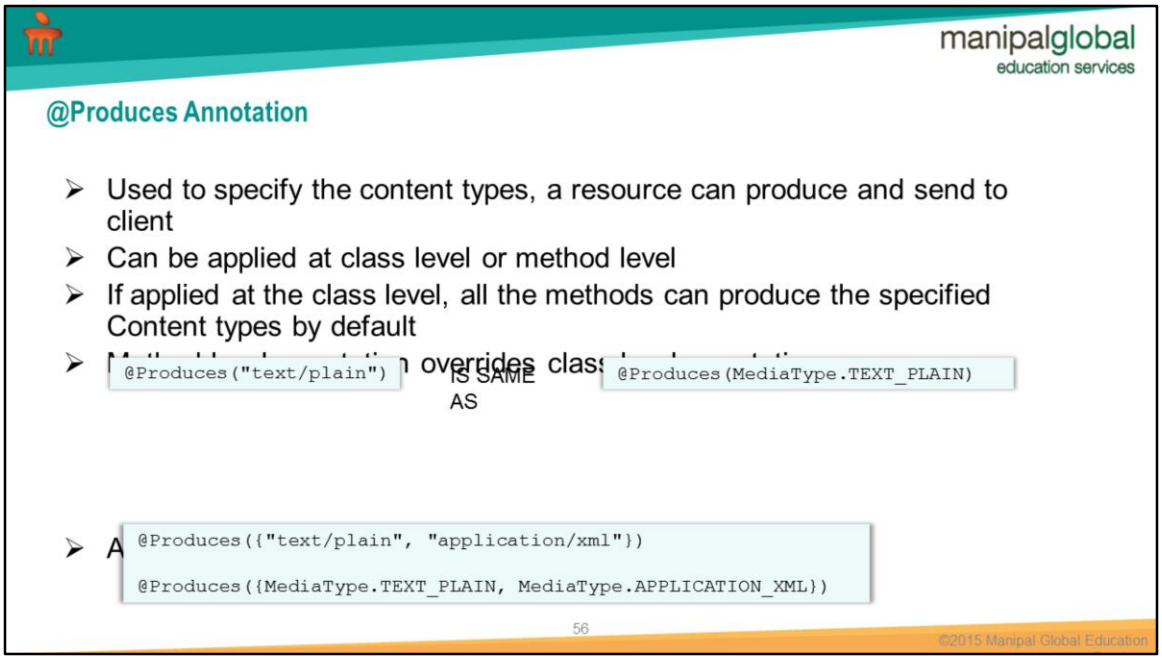

manipalglobal
education services
Request Method Designator Annotations
The behaviour of a resource is determined by the HTTP methods in the request
JAX-RS provides request method designator annotations to bind the Resource class methods to HTTP request methods
@GET
@POST
@PUT
@DELETE
@HEAD
@OPTIONS
Only one HTTP Method annotation can be applied to a single Java method
Multiple methods can have the same HTTP method annotation, but bound to different paths
JAX-RS executes the methods of resource by matching the following
HTTP Request URI with the Resource Path
54
©2015 Manipal Global Education

manipalglobal
education services

Request Method Designator Annotations

```
@Path("products")
public class ProductResource{
        ...
    @GET
    @Path("/{prodCode}")
    public Product getProduct(@PathParam("prodCode") Integer productCode){
                . . .
    }

    @DELETE
    @Path("/{prodCode}")
    public void deleteProduct(@PathParam("prodCode") Integer productCode){
                . . .
    }
}
```

- GET request to URI /products/101 will execute the getProduct(..) method
- DELETE request to URI /products/101 will execute the deleteProduct(..) method


manipalglobal
education services
@Produces Annotation
Used to specify the content types, a resource can produce and send to client
Can be applied at class level or method level
If applied at the class level, all the methods can produce the specified Content types by default
@Produces("text/plain")
IS SAME AS
@Produces(MediaType.TEXT_PLAIN)
@Produces({"text/plain", "application/xml"})
@Produces({MediaType.TEXT_PLAIN, MediaType.APPLICATION_XML})
56
©2015 Manipal Global Education

manipalglobal
education services

@Produces Annotation

```
@Path("greet")
@Produces(MediaType.TEXT_PLAIN)
public class GreetResource{

    @GET
    public String getTextGreeting(){
        return "Welcome to REST";
    }

    @GET
    @Produces(MediaType.TEXT_HTML)
    public String getHtmlGreeting(){
        return "<h1>Welcome to REST<h1>";
    }
}
```

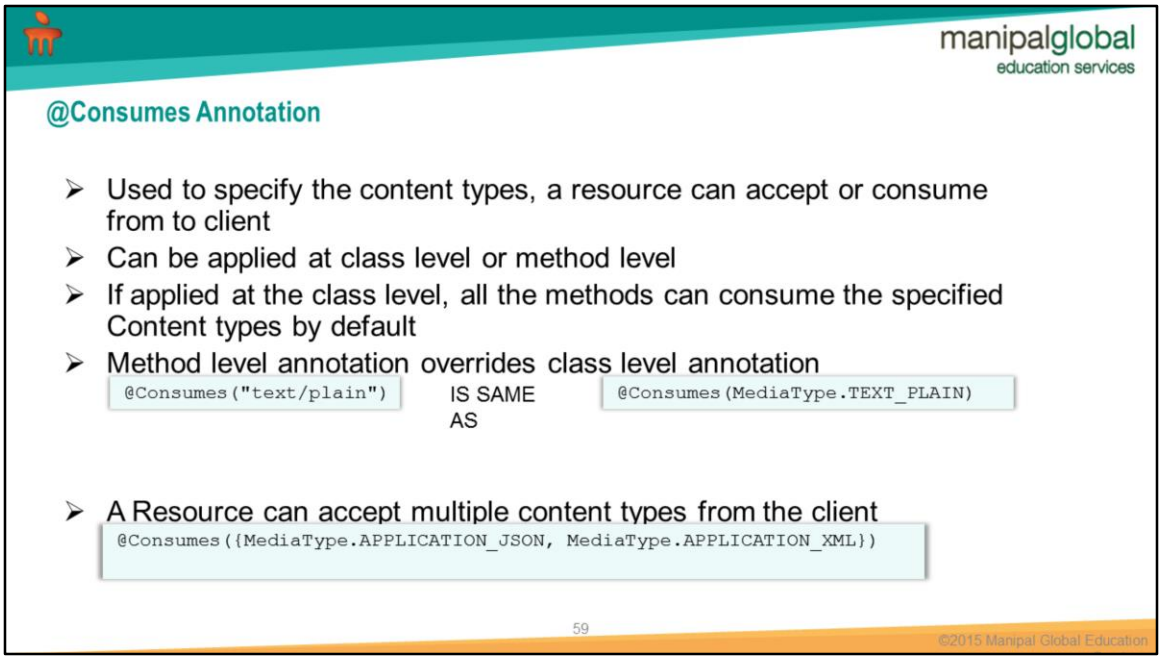
- The data returned by the method is sent in the HTTP Response Message body by JAX-RS
- JAX-RS has a bunch of built-in handlers that can marshal to and from a few different specific Java types like byte[], char[], String, File etc
- JAX-RS sets the content-type header of response based on the value of @Produces Annotation
- JAX-RS sends back an HTTP "406 Not Acceptable" response code, if no methods in the resource can produce the requested content type

manipalglobal
education services

@Produces Annotation

```
@GET
@Produces(MediaType.APPLICATION_XML)
@Path("/{prodCode}")
public Product getProduct(@PathParam("prodCode") Integer productCode){
    Product prod = service.getProduct(productCode);
    return prod;
}
```

- In the above code, the resource produces XML, but the method returns a Product Object
  - JAX-RS Implementation's have built-in JAXB handlers for classes annotated with JAXB annotations
  - JAXB handlers(Message Body Writers), marshal the Java Objects in to XML and write to the response body
- If a resource class is capable of producing more than one content type, JAX-RS chooses the content type which is acceptable by the client


manipalglobal
education services
@Consumes Annotation
Used to specify the content types, a resource can accept or consume from to client
Can be applied at class level or method level
If applied at the class level, all the methods can consume the specified Content types by default
Method level annotation overrides class level annotation
@Consumes("text/plain")
IS SAME AS
@Consumes(MediaType.TEXT_PLAIN)
A Resource can accept multiple content types from the client
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
59
©2015 Manipal Global Education

manipalglobal
education services

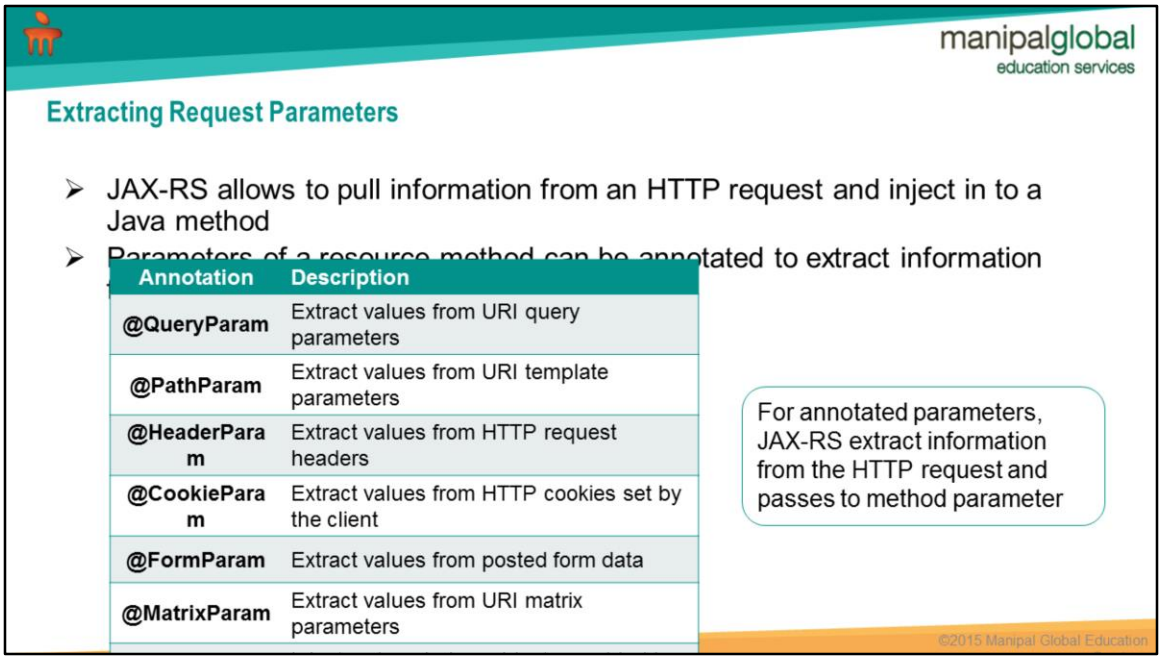
@Consumes Annotation

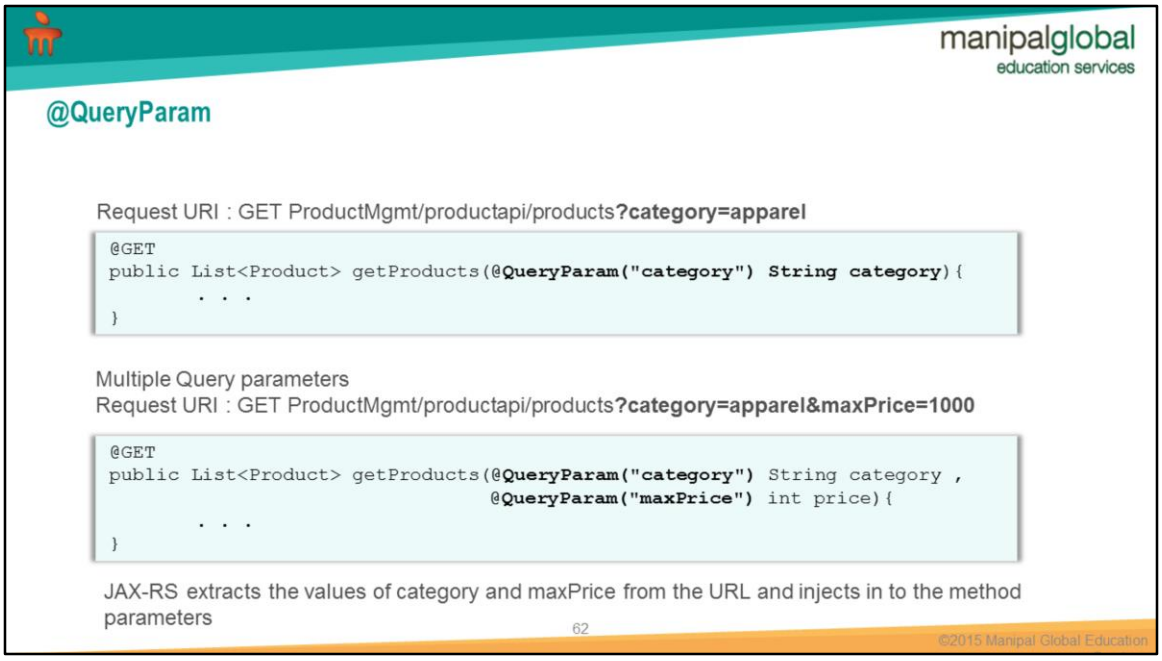
```
@POST
@Consumes(MediaType.APPLICATION_XML)
@Produces(MediaType.APPLICATION_XML)
public Product addProduct(Product prod){
    Product product = service.addProduct(prod);
    return product;
}
```

Below HTTP Request header should be added by the client, when submitting XML data

```
Content-Type: application/json
```

- In the above case, XML data sent by the client in the HTTP Request Message body is unmarshalled in to Product object by JAX-RS automatically
- JAX-RS has in built Message body readers for JAXB annotated classes and few other java data types like byte[], char[], String etc.
- JAX-RS sends back an HTTP 415 ("Unsupported Media Type") response code, if the resource does not accept the content type provided by the client


manipalglobal
education services
Extracting Request Parameters
JAX-RS allows to pull information from an HTTP request and inject in to a Java method
Parameters of a resource method can be annotated to extract information
Annotation
Description
@QueryParam
Extract values from URI query parameters
@PathParam
Extract values from URI template parameters
@HeaderParam
Extract values from HTTP request headers
@CookieParam
Extract values from HTTP cookies set by the client
@FormParam
Extract values from posted form data
@MatrixParam
Extract values from URI matrix parameters
For annotated parameters, JAX-RS extract information from the HTTP request and passes to method parameter
©2015 Manipal Global Education


manipalglobal
education services
@QueryParam
Request URI : GET ProductMgmt/productapi/products?category=apparel
@GET
public List<Product> getProducts(@QueryParam("category") String category){
. . .
}
Multiple Query parameters
Request URI : GET ProductMgmt/productapi/products?category=apparel&maxPrice=1000
@GET
public List<Product> getProducts(@QueryParam("category") String category ,
@QueryParam("maxPrice") int price){
. . .
}
JAX-RS extracts the values of category and maxPrice from the URL and injects in to the method parameters
62
©2015 Manipal Global Education

@HeaderParam and @CookieParam

```
@Path("/myservice")
public class MyService {
    @GET
    @Produces("text/html")
    public String get(@HeaderParam("Referer") String referrer,
                      @CookieParam("preferredCategory") String prefCatg) {
        ...
    }
}
```

manipalglobal
education services

@Context

@Context annotation is used to inject objects of helper classes like

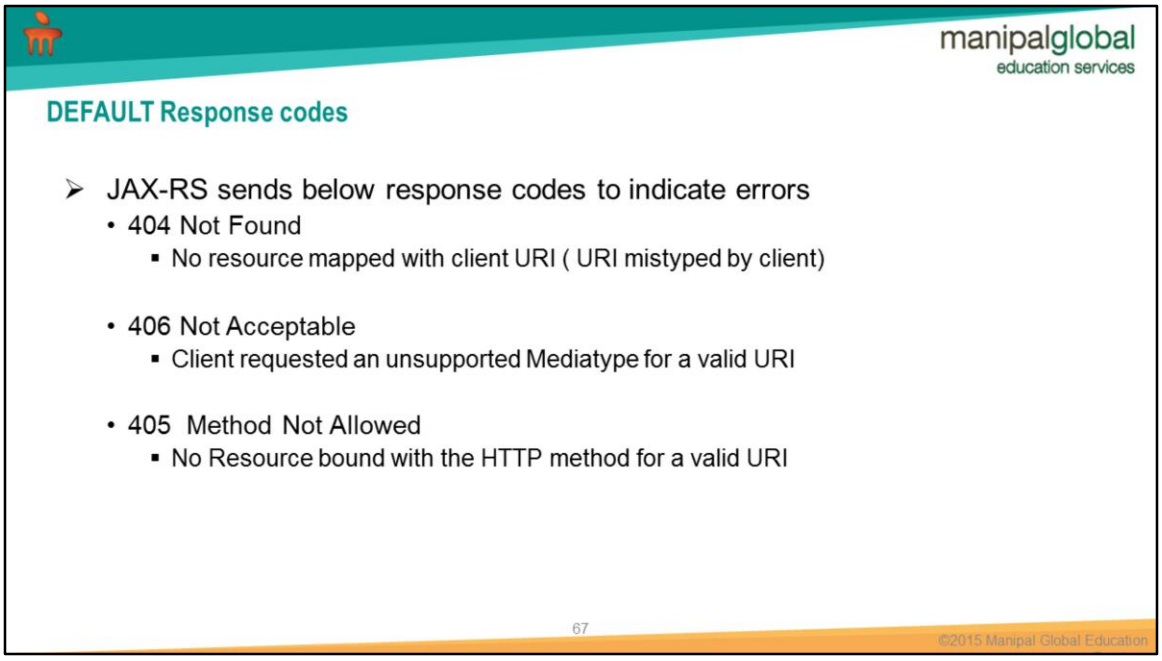
- URIInfo
- HttpHeaders
- HttpServletRequest,
- ServletContext and many more

```
@GET
@Path("uri")
@Produces(MediaType.TEXT_PLAIN)
public void getURIDetails(@Context UriInfo uri){
        String baseUri = uri.getBaseUri().toString();
        String absPath = uri.getAbsolutePath().toString();
        String path  = uri.getPath().toString();
        . . .
}
```


manipalglobal
education services
CONCEPT
Server Responses and Exception Handling
65
©2015 Manipal Global Education

DEFAULT Response codes

- JAX-RS uses some response codes as default for success and error conditions
- JAX-RS sends below response codes on successful execution of resource methods
  - 200 OK
    - Resource method returns a object which is not null
  - 204 No Content
    - Resource method returns null
    - Resource method's return type is void


manipalglobal
education services
DEFAULT Response codes
JAX-RS sends below response codes to indicate errors
404 Not Found
No resource mapped with client URI (URI mistyped by client)
406 Not Acceptable
Client requested an unsupported Mediatype for a valid URI
405 Method Not Allowed
No Resource bound with the HTTP method for a valid URI
67
©2015 Manipal Global Education

Complex Responses

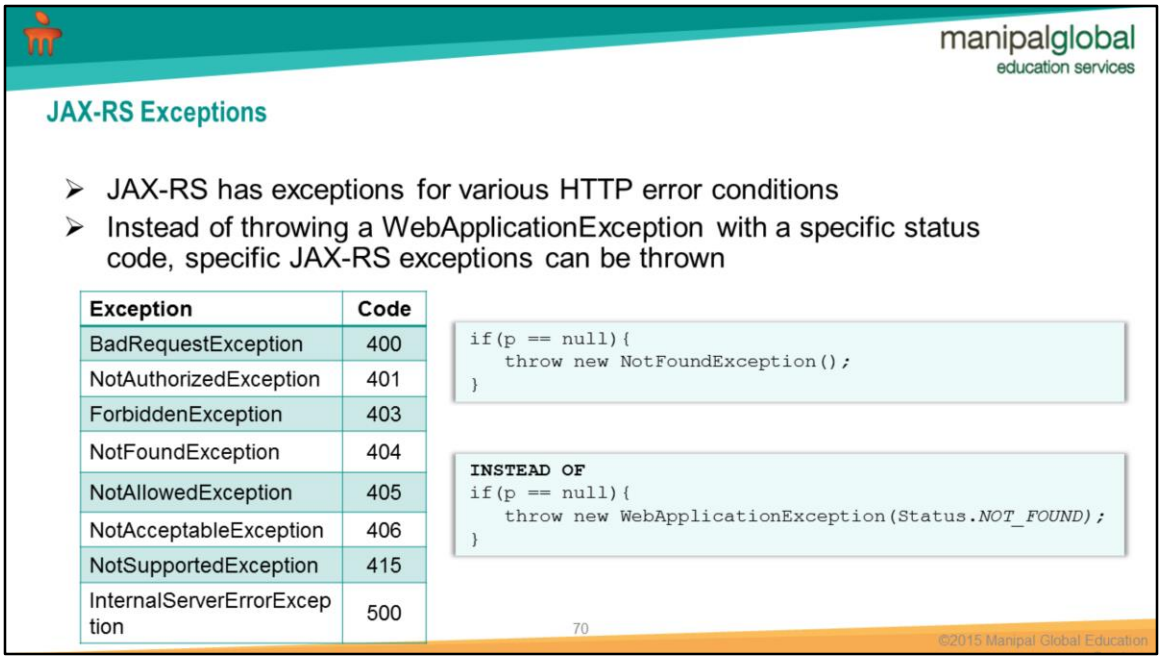
- Resources can modify the default behaviour of JAX-RS responses
- Resource's need to explicitly control the response sent, by returning instances of javax.ws.rs.core.Response

```
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response addProductUsingJSON(Product newProduct){
        Product product = service.insertProduct(newProduct);
        ResponseBuilder builder = Response.status(Status.CREATED);
        builder.entity(product)
               .header("some-header", "some-value");
        Response response = builder.build();
        return response;
}
```

Exception Handling

- Errors can be reported to a client either by returning the appropriate Response object or by throwing an exception
- WebApplicationException provided by JAX-RS can be thrown by application code and automatically processed by JAX-RS

```
@DELETE
@Path("/{prodCode}")
public void removeProduct(@PathParam("prodCode") Integer prodCode){
        Product p = service.deleteProduct(prodCode);
        if(p == null){
                throw new WebApplicationException(Status.NOT_FOUND);
        }
}
```


manipalglobal
education services
JAX-RS Exceptions
JAX-RS has exceptions for various HTTP error conditions
Instead of throwing a WebApplicationException with a specific status code, specific JAX-RS exceptions can be thrown
Exception
Code
BadRequestException
400
NotAuthorizedException
401
ForbiddenException
403
NotFoundException
404
NotAllowedException
405
NotAcceptableException
406
NotSupportedException
415
InternalServerErrorException
500
if(p == null){
throw new NotFoundException();
}
INSTEAD OF
if(p == null){
throw new WebApplicationException(Status.NOT_FOUND);
}
70
©2015 Manipal Global Education


manipalglobal
education services
CONCEPT
JAX-RS Web
Service Client
71
©2015 Manipal Global Education

JAX-RS Client API

- The JAX-RS Client API provides a high-level API for accessing any REST resources
- Client API is defined in the javax.ws.rs.client package

- Following steps are needed to access a REST resource using the Client API
  - Obtain an instance of the javax.ws.rs.client.Client interface
  - Configure the Client instance with a target
  - Create a request based on the target
  - Invoke the request and get the Response

manipalglobal
education services

Creating JAX-RS Client

package:
mgait.client
Class :

- Client API is designed to be fluent
- Request is configured and invoked by chaining method calls in few lines of code

```
Client client = ClientBuilder.newClient();
WebTarget greet = client.target("http://buy.com/ProductMgmt/prodapi/greet");
String greeting = greet
                  .request(MediaType.TEXT_PLAIN)
                  .get(String.class);
client.close();
```

- Client instance is first created by calling the ClientBuilder.newClient() method
- Clients target() method sets the target of the client based on a URI
- WebTarget's request method sets the MediaType acceptable for the response
- get method invokes the service using an HTTP GET request and sets the type of the returned entity to String

manipalglobal
education services

Setting Client Target

- The target of a client is a REST resource at a particular URI
- Target is represented by an instance of the WebTarget interface

```
Client client = ClientBuilder.newClient();
WebTarget prodTraget =
                client.target("http://buy.com/ProductMgmt/prodapi/products/101");
```

- For complex Rest resources with multiple targets, several targets can be constructed by building a base WebTarget

```
WebTarget base =  client.target("http://buy.com/ProductMgmt/prodapi");

WebTarget greet = base.path("greet");
WebTarget products = base.path("products");
WebTarget product = base.path("products/101");
```

manipalglobal
education services

Setting Path Parameters and QueryParameters

- Path parameters in client requests can be specified as URI template parameters
- Template parameters are specified by surrounding the template variable

```
WebTarget base =  client.target("http://buy.com/ProductMgmt/prodapi")
WebTarget product =  base.path("products/{prodCode}");

WebTarget prod101 = product.resolveTemplate("prodCode", "101");
WebTarget prod102 = product.resolveTemplate("prodCode", "102");


WebTarget electronics = base.path("products")
                        .queryParam("category", "electronics");
```

manipalglobal
education services

Invoking GET Requests

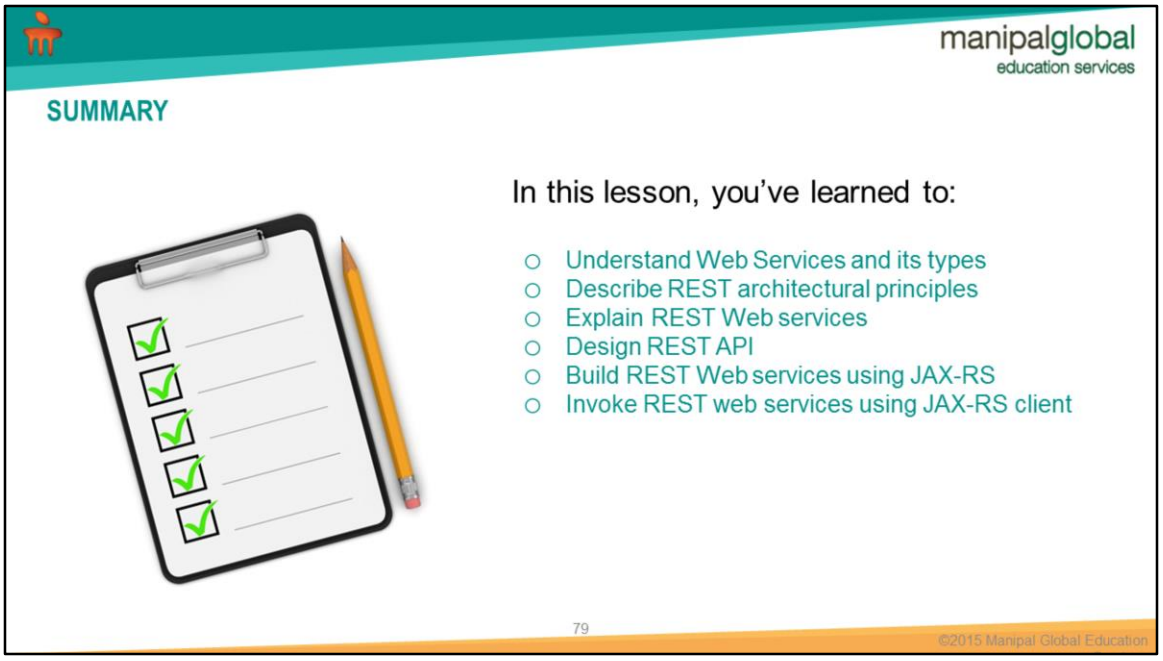
```
WebTarget prod101 = client.target("http://buy.com/ProductMgmt/prodapi/products/101");

Response resp1 = prod101
                .request(MediaType.APPLICATION_JSON)
                .get();

if(resp1.getStatus() == 200){
        Product prod1 = resp.readEntity(Product.class);
        System.out.println(prod2.getProductName());
}

Alternatively

Product prod2 = prod101
                .request(MediaType.APPLICATION_JSON)
                .get(Product.class);
```

manipalglobal
education services

Invoking Post requests

```
WebTarget baseTarget = client.target("http://localhost:8686/ProductMgmt/productapi");
WebTarget productsTarget = baseTarget.path("products");

Product newProduct = new Product();
newProduct.setCategory("electronics");
newProduct.setProductName("Mobile");
newProduct.setPrice(15000);

Product addedProduct = productsTarget
               .request(MediaType.APPLICATION_XML)
               .post(Entity.xml(newProduct), Product.class);

System.out.println(addedProduct.getProductCode());
```


manipalglobal
education services
SUMMARY
REST Web Services
78
©2015 Manipal Global Education


manipalglobal
education services
SUMMARY
In this lesson, you've learned to:
Understand Web Services and its types
Describe REST architectural principles
Explain REST Web services
Design REST API
Build REST Web services using JAX-RS
Invoke REST web services using JAX-RS client
79
©2015 Manipal Global Education