

# Session 12

## RESTful Services

1

## Lecture Objectives

- Understand the fundamental concepts of Web services
- Become familiar with JAX-RS annotations
- Be able to build a simple Web service

© Robert Kelly, 2018

2

## Reading & References

### ■ Reading

#### ■ Tutorials

[https://javabrainz.io/courses/javaee\\_jaxrs/](https://javabrainz.io/courses/javaee_jaxrs/)  
[docs.oracle.com/javaee/7/tutorial/webservices-intro.htm#GIJTI](https://docs.oracle.com/javaee/7/tutorial/webservices-intro.htm#GIJTI)  
(Chapters 27 and 29.1-29.3)

### ■ Reference

#### ■ Java EE API

[docs.oracle.com/javaee/7/api/javax/ws/rs/package-summary.html](https://docs.oracle.com/javaee/7/api/javax/ws/rs/package-summary.html)

#### ■ Book

RESTful Java Web Services, 3<sup>rd</sup> Edition,  
<https://www.amazon.com/RESTful-Java-Web-Services-pragmatic/dp/1788294041>

*Be careful - other JAX-RS documentation assumes knowledge of other Java EE technologies (e.g., JPA)*

*Session material follows Java EE 7 Tutorial text*

## Client - Servlet Model

- Requires logic in servlet to route each request to a service method
- Does not directly use URL and other http data to route to a service
- Mapping of the URL to a servlet is handled with web.xml or Java Annotation in servlet class

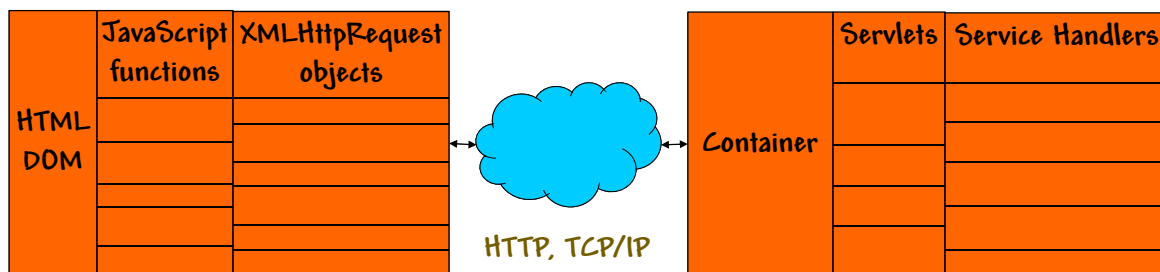
*Java Annotation enables a more flexible approach to mapping requests to services*

```
<form method="get" action=  
"http://localhost:8080/CSE336-2017/helloyou.html">
```

*Servlet identified by the "helloyou.html" URL string usually acts as a controller, and routes to a service handler*

## Client/Server Interaction

Our definition of client/server interaction to date



Parts of the client/server interaction are abstracted by tools/libraries

© Robert Kelly, 2018

5

## RESTful Web Services

- Representational State Transfer
- Architectural style for distributed systems
- Architecturally consistent with http
- Provides a standard means of interoperating between software applications running on a variety of platforms and frameworks
- Use existing W3C and IETF standards (HTTP, XML, URI, MIME)

A service is a software component provided through a network-accessible endpoint

© Robert Kelly, 2018

6

## Types of Web Services

### ■ JAX-WS

- Communication using XML
- Provides for message-oriented and RPC services
- Uses SOAP messages
- includes standards for security and reliability

### ■ JAX-RS

- Standard
- Semantics of the data to be exchanged is understood by client and server

© Robert Kelly, 2018

7

## JAX-RS

### ■ Java API for RESTful Web Services

### ■ A standard - not a product

### ■ Reference implementations

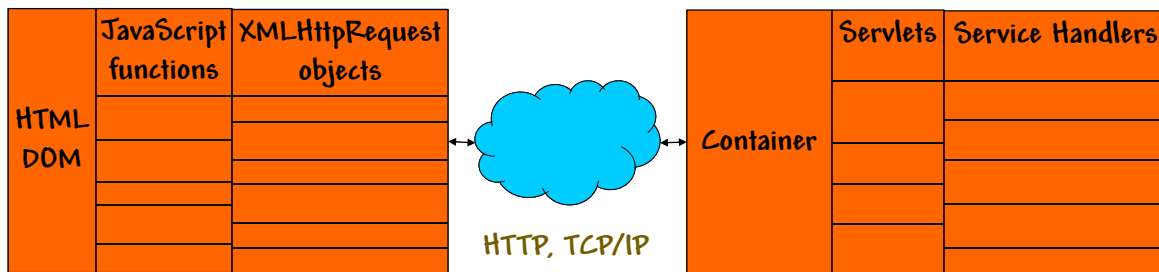
- Jersey, RESTeasy, et al, along with some application servers
- No requirement to implement on top of servlets, but many implementation do

© Robert Kelly, 2018

8

## Client/Server Interaction

Think of JAX-RS as extending the abstraction to the Service handlers



Parts of the client/server interaction are abstracted by tools/libraries

© Robert Kelly, 2018

9

## Principles of REST Architectural Style

- Resource identification through URI
- Uniform interface - CRUD access defined in HTTP methods (PUT, GET, POST, and DELETE)
- Self-descriptive messages - content can be accessed in a variety of formats (e.g., HTML, XML, plain text, PDF, JPEG, JSON, etc.). Metadata about the resource is available
- Stateful interactions through links - Interactions are stateless (request messages contain state info)

CRUD = Create, Read, Uppdate, and Delete

© Robert Kelly, 2018

10

## Implications of REST Style

- Interactions are predominantly computer-computer, not human-computer
- Resource based URI
- Typically published as an API, so design and URI naming important
- Expanded and more precise use of http methods
- Expanded use of http status codes
- Content negotiation between client and server

URI requests are usually nouns, not verbs

© Robert Kelly, 2018

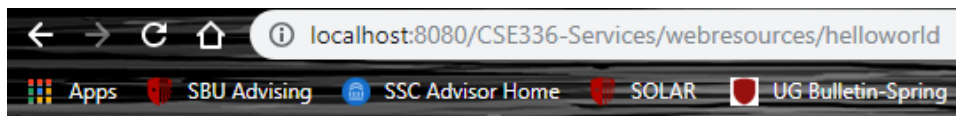
11

## Example

- We start by building a very simple RESTful service
- In the next session, we will extend this by

- Passing parameters to the server
- Negotiating content
- Returning content

For all the examples, think of accessing the resources from your html/JavaScript running in your browser



**Hello, World!!**

© Robert Kelly, 2018

12

## Creating a RESTful Root Resource Class

- Root resource classes are POJOs (plain old Java objects)
- Annotated with `@Path` or a request method designator (`@GET`, `@PUT`, `@POST`, or `@DELETE`)

JAX-RS uses Java Annotations

© Robert Kelly, 2018

13

## JAX-RS Annotation Summary ...

Annotation	Description
<code>@PATH</code>	Relative URI indicating where the class will be hosted. Can also embed variables (e.g., <code>/helloworld/{username}</code> )
<code>@GET</code>	Corresponds to the HTTP GET method. A Java method annotated with <code>@GET</code> will handle GET requests
<code>@POST</code>	Corresponds to the HTTP POST method. Intended for new resources.
<code>@PUT</code>	Corresponds to HTTP PUT method. Intended for resource updates
<code>@DELETE</code>	Corresponds to HTTP DELETE method

© Robert Kelly, 2018

14

## ... JAX-RS Annotation Summary

Annotation	Description
@HEAD	Corresponds to HTTP Method.
@PathParam	Parameter extracted from the request URI. Parameter names correspond to the URI path template variable names specified in the @PATH annotation
@QueryParam	Extracted from the query string
@Consumes	Specifies the MIME type sent by client
@Produces	Specifies the MIME type produced (e.g., "text/plain")
@ApplicationPath	Defines the URL mapping. Base URI for all resource URIs specified by @Path

© Robert Kelly, 2018

15

## Web Services With NetBeans ...

### Create a new project (or use an existing one)

The screenshot shows the 'Choose Project' dialog in NetBeans. On the left, under 'Categories', 'Java Web' is selected. On the right, under 'Projects', 'Web Application' is selected. Below the dialog, the 'Steps' section lists: 1. Choose Project, 2. Name and Location, 3. Server and Settings, 4. Frameworks. The 'Name and Location' section shows: Project Name: CSE336-Services, Project Location: C:\Users\robkelly\Documents\NetBeansProjects, and Project Folder: C:\Users\robkelly\Documents\NetBeansProjects\CSE336-Services.

© Robert Kelly, 2018

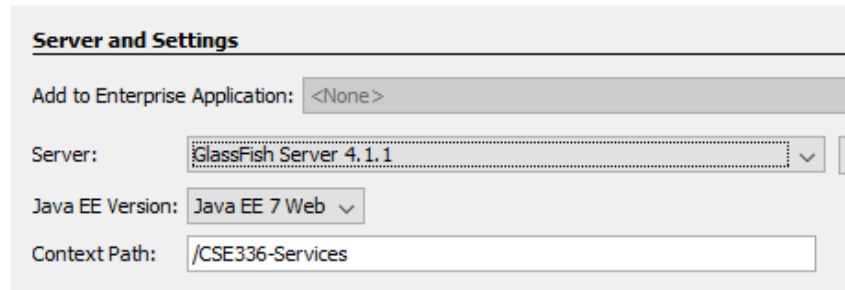
16



## ... Web Services With NetBeans ...

- Set libraries
- No need to declare any frameworks

Note that Glassfish includes a reference implementation of JAX-RS



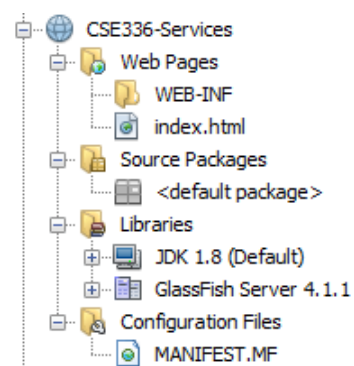
© Robert Kelly, 2018

17

## ... Web Services With NetBeans ...

- You are now set to define your first JAX-RS application

Initial application folder

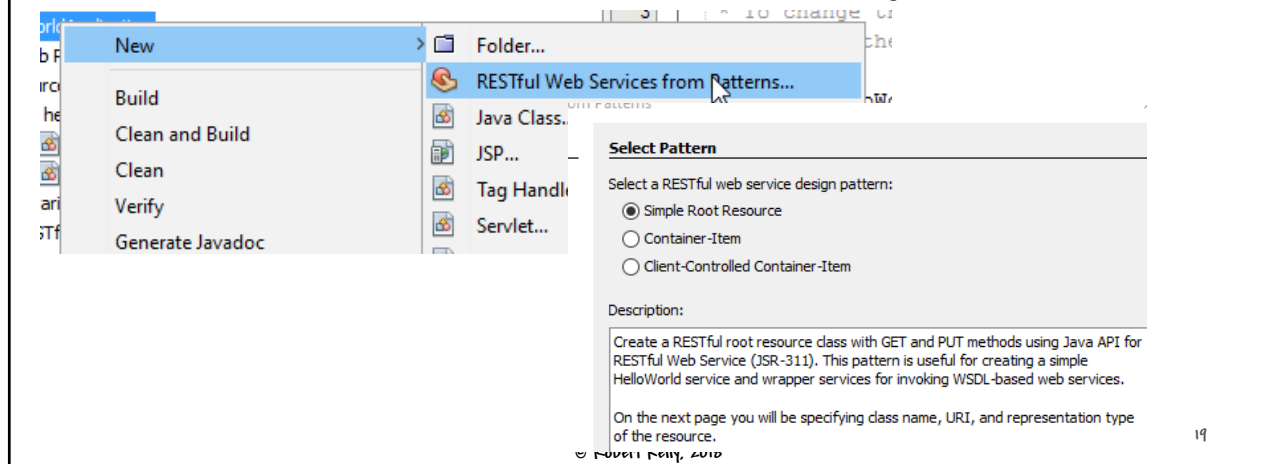


© Robert Kelly, 2018

18

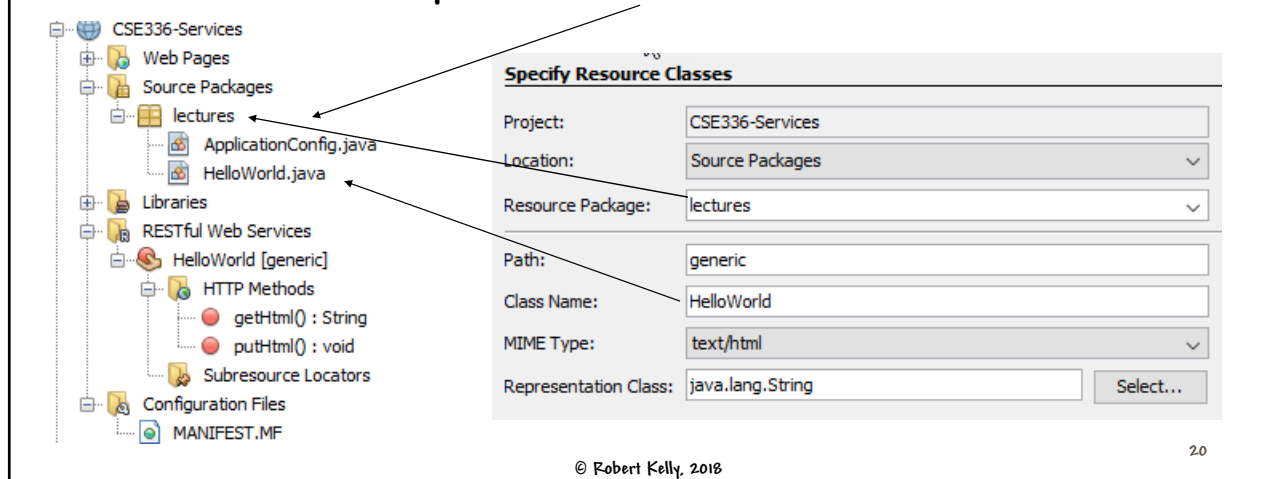
## ... Web Services With NetBeans ...

- NetBeans includes a feature to create a new RESTful Web Service
- Start to create a helloworld application with a right click on project



## ... Web Services With NetBeans

- Specify resource class
- NetBeans will set up with some starter code



## ApplicationConfig Class

Registers the classes of the application with the JAX-RS implementation

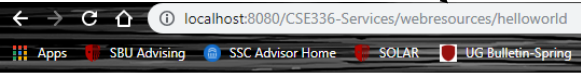
```

package helloWorld;
import java.util.Set;
import javax.ws.rs.core.Application;
@javax.ws.rs.ApplicationPath("webresources")
public class ApplicationConfig extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }
    /** Do not modify addRestResourceClasses() method.
     * It is automatically populated with all resources
     * defined in the project. You may comment out this method call in getClasses().
     */
    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(helloWorld.HelloWorld.class);
    }
}
    
```

This config file is automatically generated by NetBeans

The ApplicationPath serves as the base URL to locate the services

You can also specify the base URI in the web.xml



Hello, World!!

© Robert Kelly, 21

## HelloWorld.java

```

@Path("helloworld")
public class HelloWorld {
    @Context
    private UriInfo context;

    public HelloWorld() {
    }
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String getHtml() {
        return "<html><body><h1>Hello, World!!</body></html>";
    }
    @PUT
    @Consumes(MediaType.TEXT_HTML)
    public void putHtml(String content) {
    }
}
    
```

Import and package statements not shown

Path relative to the URI path defined with ApplicationPath annotation

Identifies the MIME type of the response

An http GET request will return the html

© Robert Kelly, 2018 22

## MediaType Class

- `javax.ws.rs.core.MediaType`
- An abstraction for JAX-RS media types
- Contains String constants
- Examples
  - `TEXT_HTML` - "text/html"
  - `TEXT_PLAIN` - "text/plain"
  - `APPLICATION_JSON` - "application/json"

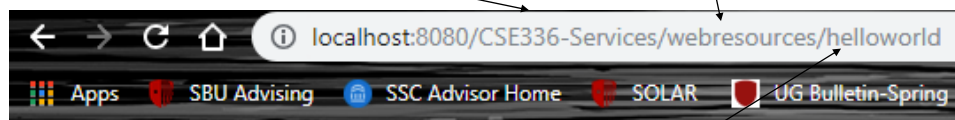
© Robert Kelly, 2018

23

## Test the Web Service

- Start the application, then access your services through your browser
- Note the URL
  - Application (or project)

Specified in `ApplicationConfig`



### Hello, World!!

Without a form, the http request is likely a GET

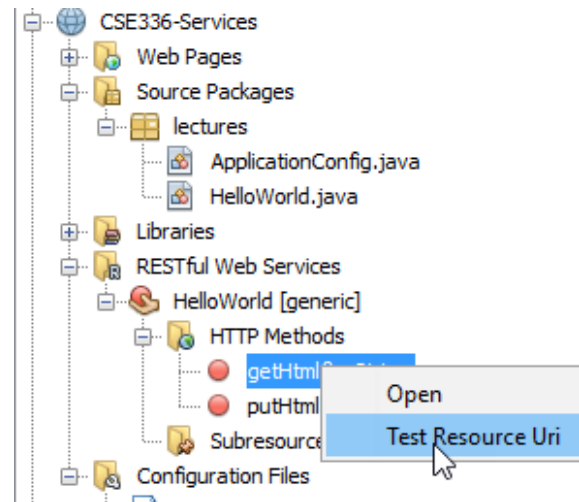
Specified with `@Path` in `HelloWorld` class

© Robert Kelly, 2018

24

## Test the Service in NetBeans

- Right click on service method.
- Response opens in default browser



© Robert Kelly, 2018

25

## Available Web Services

- Google Maps
- flickr
- Yahoo News Search
- YouTube
- Facebook
- Lots more

<http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=finances&format=pdf>

The above URL will search (maybe?) the Web database for PDF files containing the term "finances"

© Robert Kelly, 2018

26

## Are We On Track?

- Verify that you can build a web service in your IDE
- Use the same example as shown in the slides

© Robert Kelly, 2018

27

## Assignment

- Use your Brooklyn Library html to make web service calls on a library resource
- Define a Java class whose data includes all the items in the Brooklyn Library form (plus a card # field).
- The data structure in the class should hold some collection of library card data
- When instantiated, the card data structure will initialize to 5 cards
- Write a GET service to retrieve card data when a user enters the card # and last name. If combo is found, returned data will populate the form.
- Write a POST service to add form data to the data structure when the Submit button is pressed
- Use a XMLHttpRequest to send the request to the service

© Robert Kelly, 2018

28

## Have You Achieved the Lecture Objectives?

- Understand the fundamental concepts of Web services
- Become familiar with JAX-RS annotations
- Be able to build a simple Web service