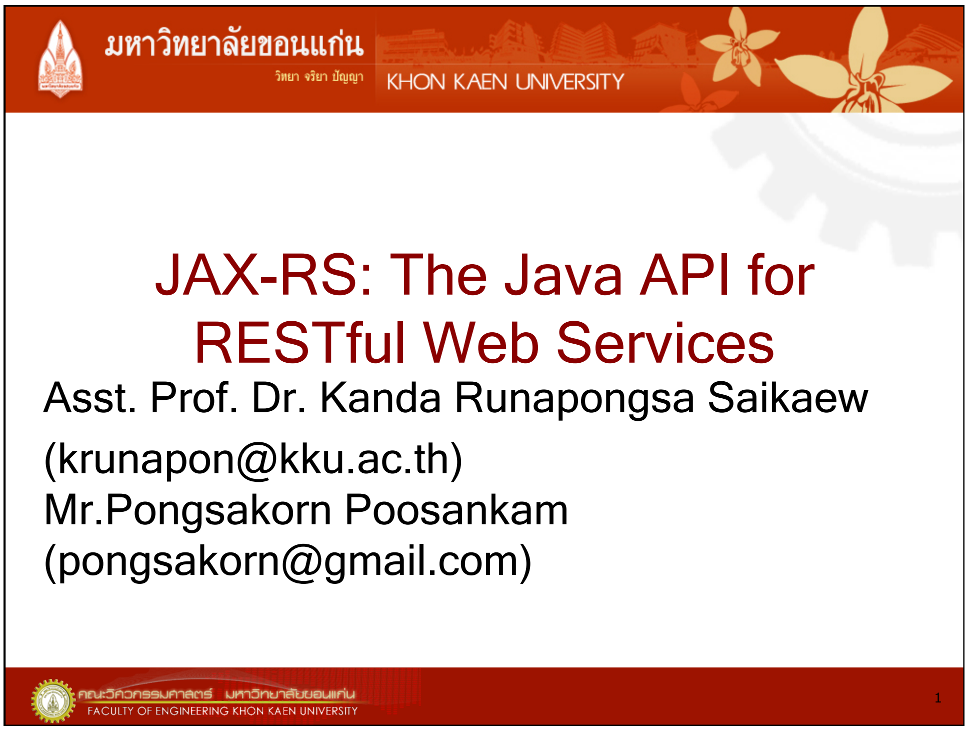
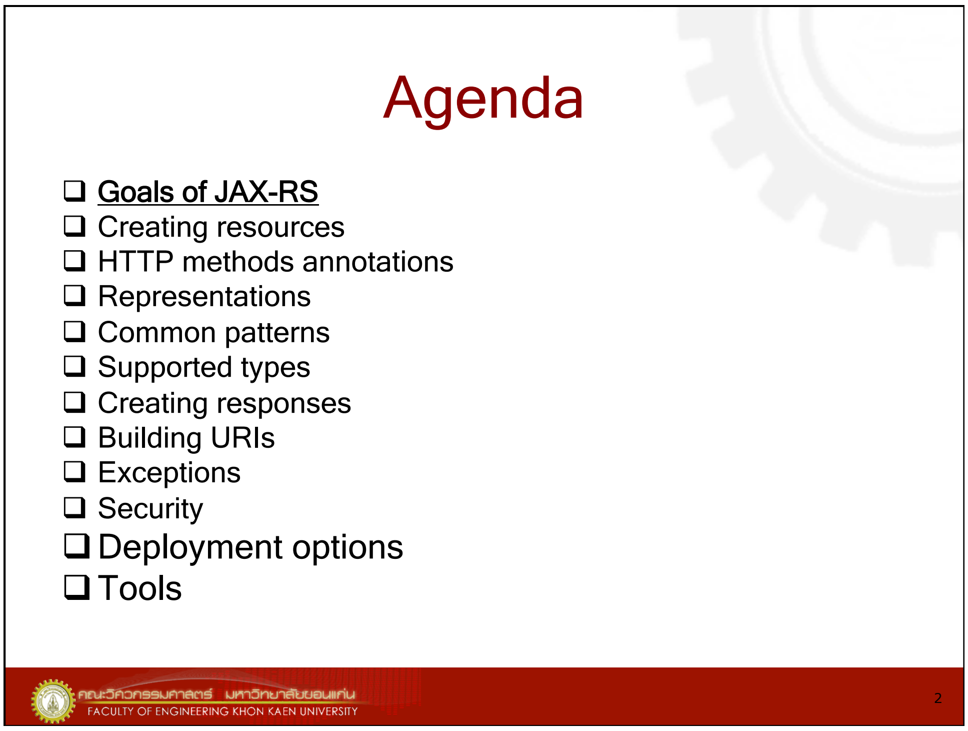


มหาวิทยาลัยขอนแก่น
วิทยา จริยา ปัญญา
KHON KAEN UNIVERSITY

JAX-RS: The Java API for RESTful Web Services

Asst. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Mr.Pongsakorn Poosankam
(pongsakorn@gmail.com)

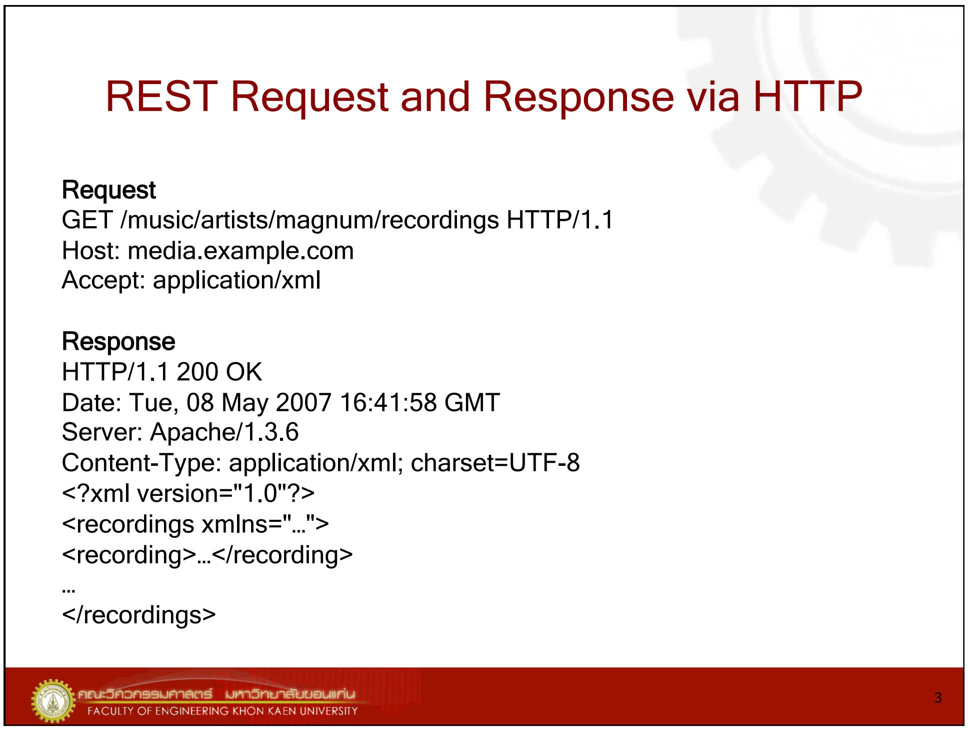
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
1



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools

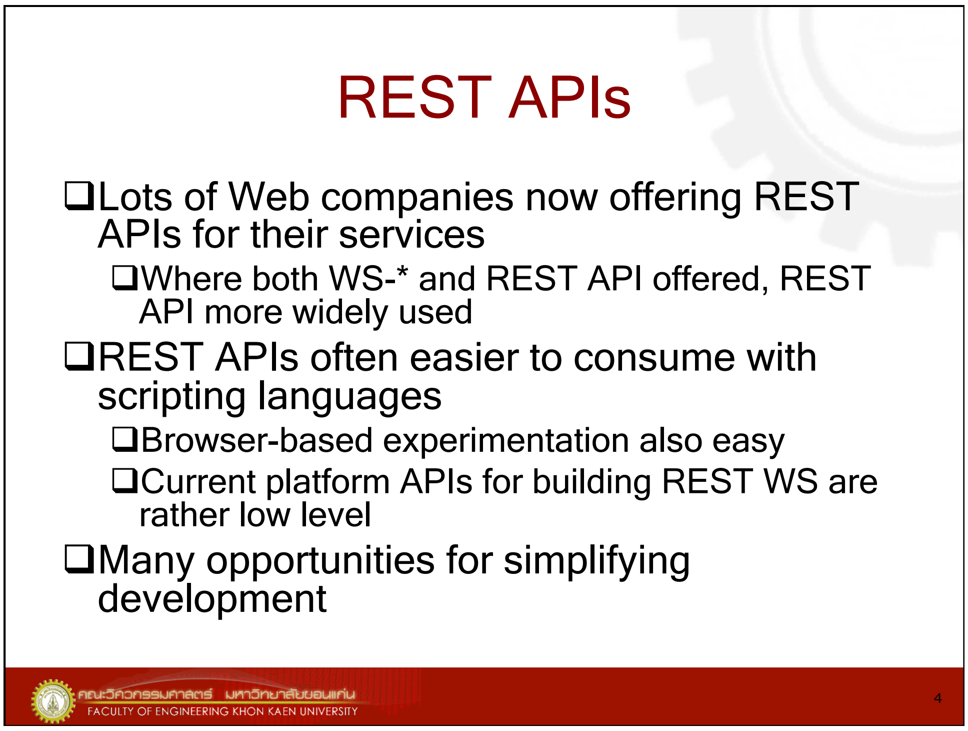
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY
2



REST Request and Response via HTTP

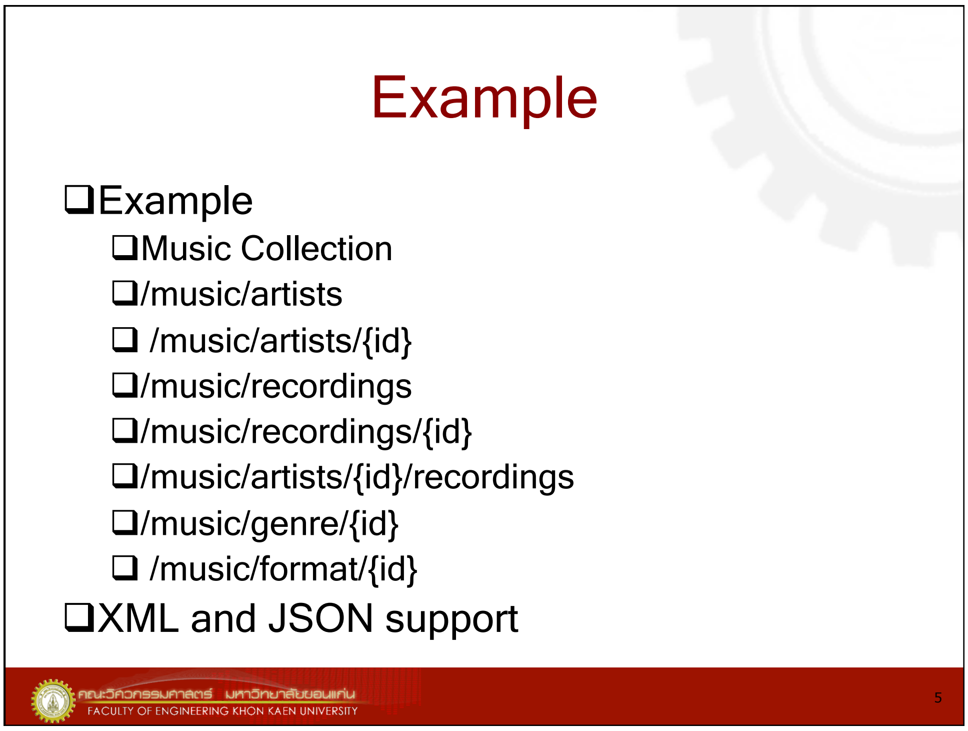
Request
```
GET /music/artists/magnum/recordings HTTP/1.1
Host: media.example.com
Accept: application/xml
```

Response
```
HTTP/1.1 200 OK
Date: Tue, 08 May 2007 16:41:58 GMT
Server: Apache/1.3.6
Content-Type: application/xml; charset=UTF-8
<?xml version="1.0"?>
<recordings xmlns="…">
<recording>…</recording>
…
</recordings>
```



REST APIs

- ❑Lots of Web companies now offering REST APIs for their services
  - ❑Where both WS-* and REST API offered, REST API more widely used
- ❑REST APIs often easier to consume with scripting languages
  - ❑Browser-based experimentation also easy
  - ❑Current platform APIs for building REST WS are rather low level
- ❑Many opportunities for simplifying development

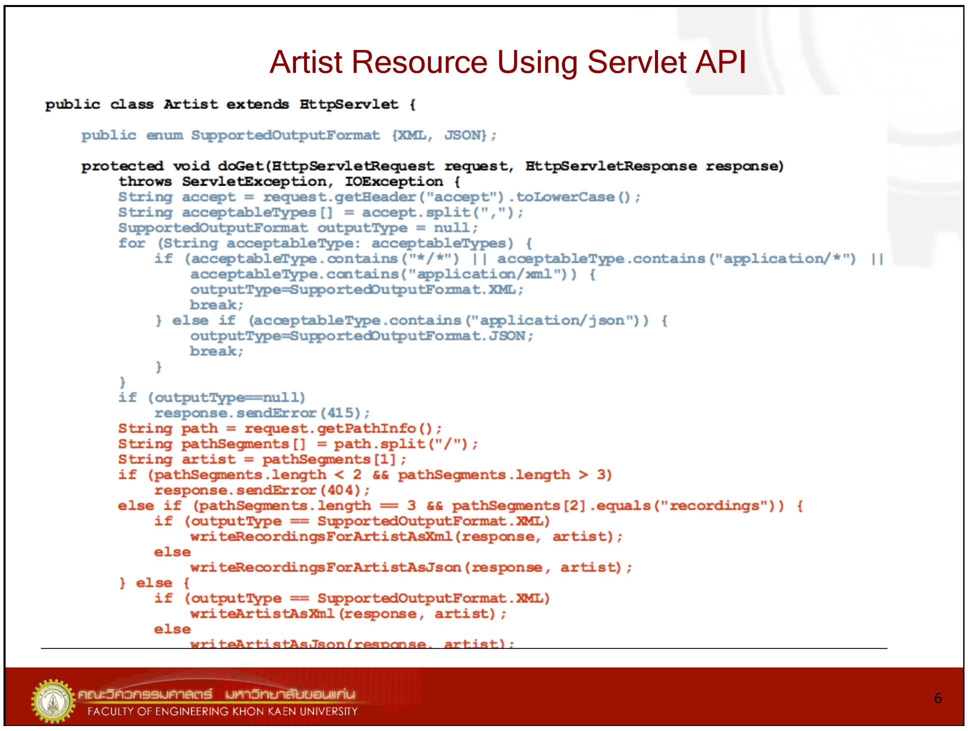


Example

- ❑Example
  - ❑Music Collection
  - ❑/music/artists
  - ❑ /music/artists/{id}
  - ❑/music/recordings
  - ❑/music/recordings/{id}
  - ❑/music/artists/{id}/recordings
  - ❑/music/genre/{id}
  - ❑ /music/format/{id}
- ❑XML and JSON support

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

5

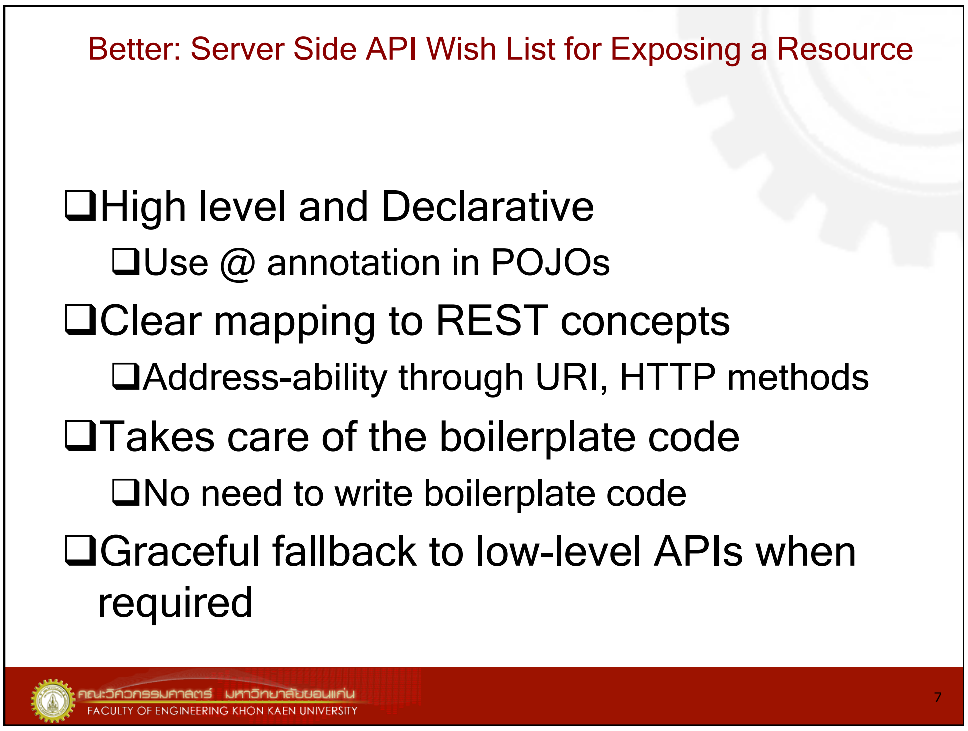


Artist Resource Using Servlet API

```
public class Artist extends HttpServlet {

    public enum SupportedOutputFormat {XML, JSON};

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accept = request.getHeader("accept").toLowerCase();
        String acceptableTypes[] = accept.split(",");
        SupportedOutputFormat outputType = null;
        for (String acceptableType: acceptableTypes) {
            if (acceptableType.contains("*/*") || acceptableType.contains("application/*") ||
                acceptableType.contains("application/xml")) {
                outputType=SupportedOutputFormat.XML;
                break;
            } else if (acceptableType.contains("application/json")) {
                outputType=SupportedOutputFormat.JSON;
                break;
            }
        }
        if (outputType==null)
            response.sendError(415);
        String path = request.getPathInfo();
        String pathSegments[] = path.split("/");
        String artist = pathSegments[1];
        if (pathSegments.length < 2 && pathSegments.length > 3)
            response.sendError(404);
        else if (pathSegments.length == 3 && pathSegments[2].equals("recordings")) {
            if (outputType == SupportedOutputFormat.XML)
                writeRecordingsForArtistAsXml(response, artist);
            else
                writeRecordingsForArtistAsJson(response, artist);
        } else {
            if (outputType == SupportedOutputFormat.XML)
                writeArtistAsXml(response, artist);
            else
                writeArtistAsJson(response, artist);
```

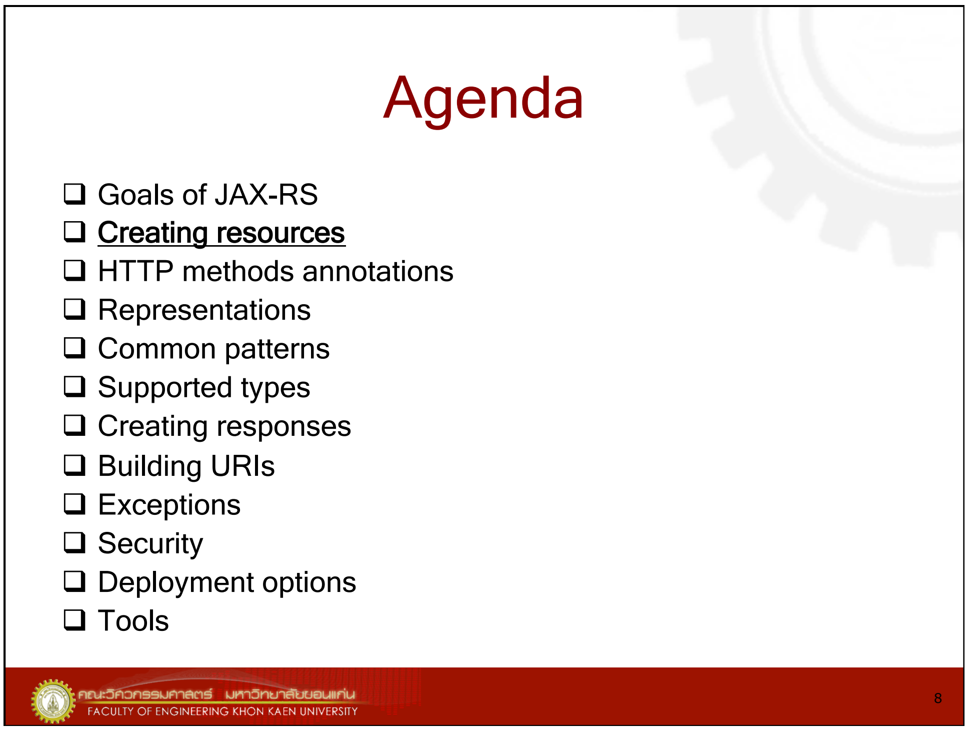
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

6



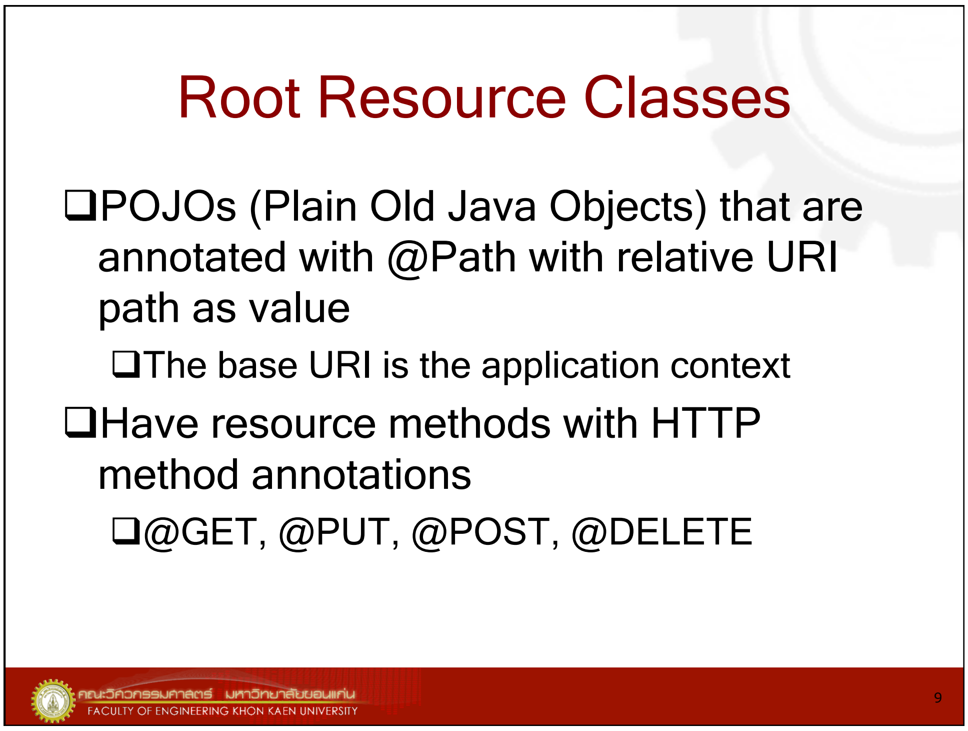
Better: Server Side API Wish List for Exposing a Resource

- ❑High level and Declarative
  - ❑Use @ annotation in POJOs
- ❑Clear mapping to REST concepts
  - ❑Address-ability through URI, HTTP methods
- ❑Takes care of the boilerplate code
  - ❑No need to write boilerplate code
- ❑Graceful fallback to low-level APIs when required



Agenda

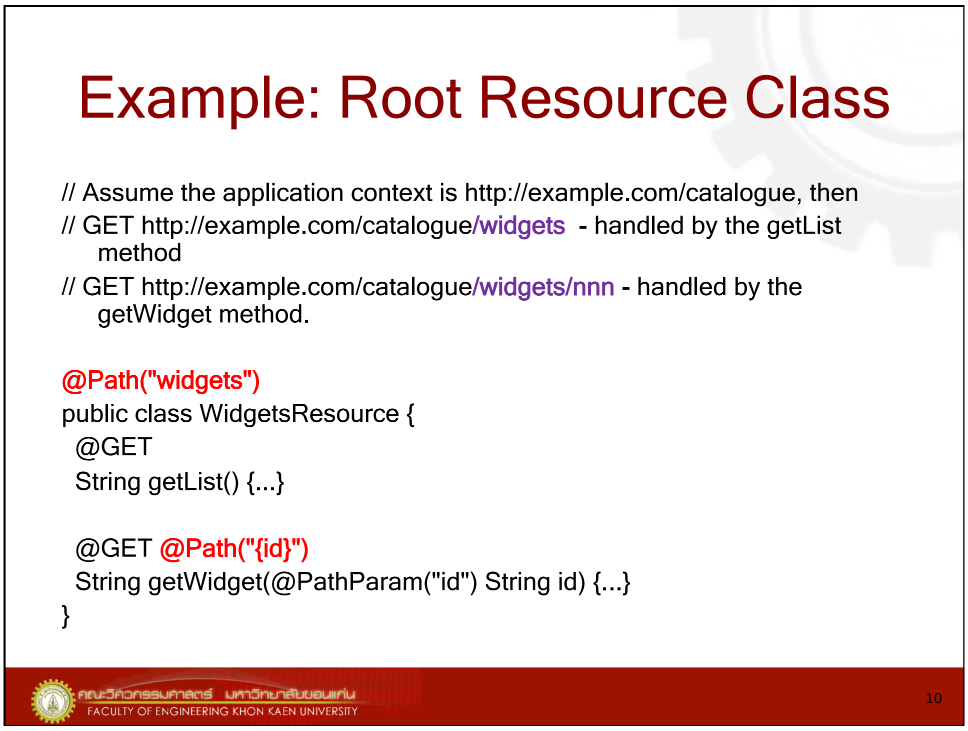
- ❑ Goals of JAX-RS
- ❑ **Creating resources**
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools



Root Resource Classes

- ❑POJOs (Plain Old Java Objects) that are annotated with @Path with relative URI path as value
  - ❑The base URI is the application context
- ❑Have resource methods with HTTP method annotations
  - ❑@GET, @PUT, @POST, @DELETE

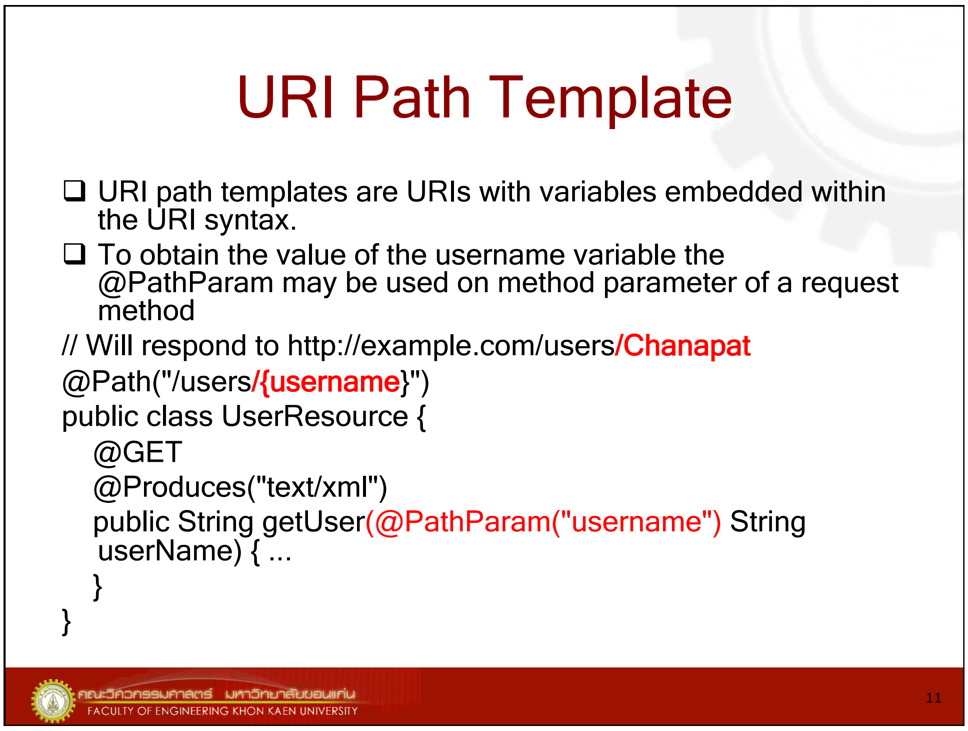
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 9



Example: Root Resource Class

```
// Assume the application context is http://example.com/catalogue, then
// GET http://example.com/catalogue/widgets  - handled by the getList
    method
// GET http://example.com/catalogue/widgets/nnn - handled by the
    getWidget method.

@Path("widgets")
public class WidgetsResource {
  @GET
  String getList() {...}

  @GET @Path("{id}")
  String getWidget(@PathParam("id") String id) {...}
}
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 10



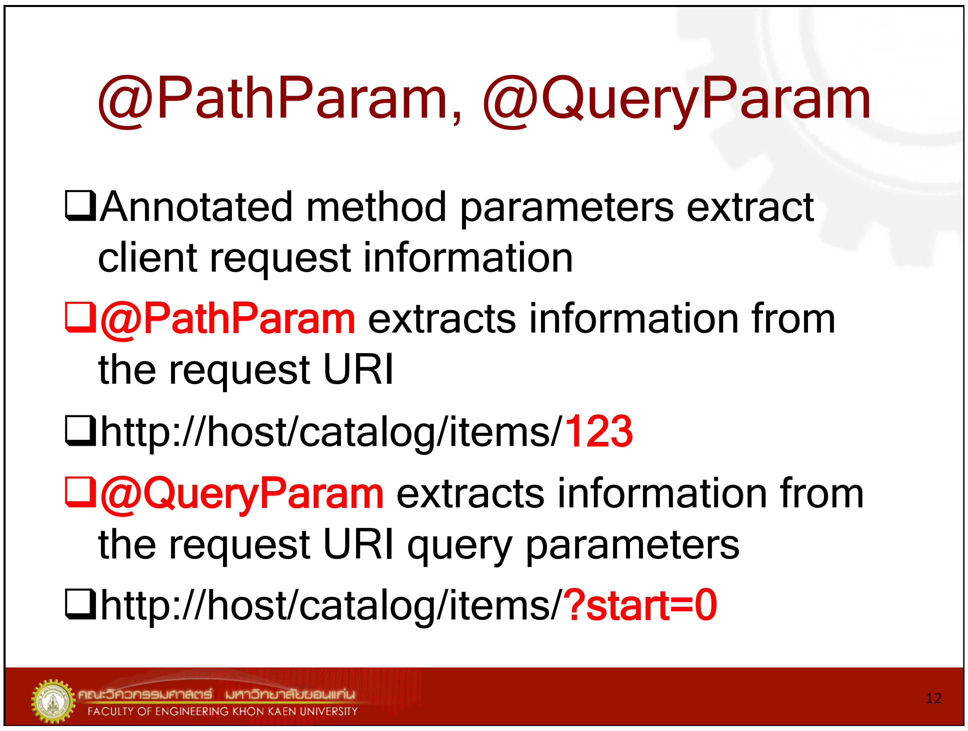
URI Path Template

- ❑ URI path templates are URIs with variables embedded within the URI syntax.
- ❑ To obtain the value of the username variable the @PathParam may be used on method parameter of a request method

```
// Will respond to http://example.com/users/Chanapat
@Path("/users/{username}")
public class UserResource {
   @GET
   @Produces("text/xml")
   public String getUser(@PathParam("username") String
    userName) { ...
   }
}
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

11

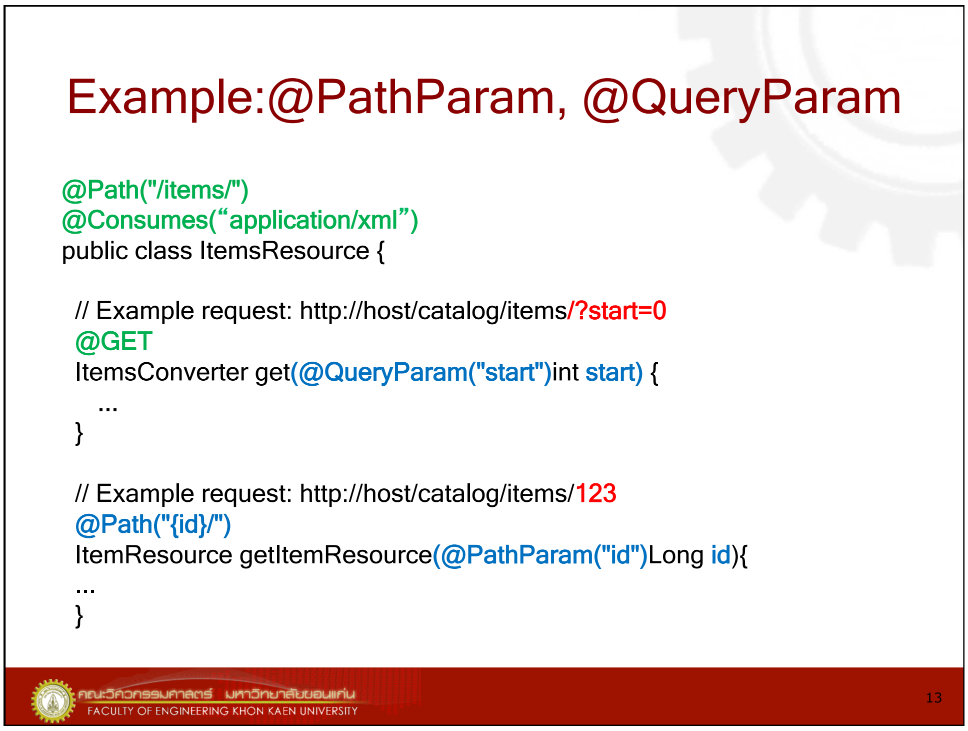


@PathParam, @QueryParam

- ❑ Annotated method parameters extract client request information
- ❑ **@PathParam** extracts information from the request URI
- ❑ http://host/catalog/items/**123**
- ❑ **@QueryParam** extracts information from the request URI query parameters
- ❑ http://host/catalog/items/**?start=0**

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

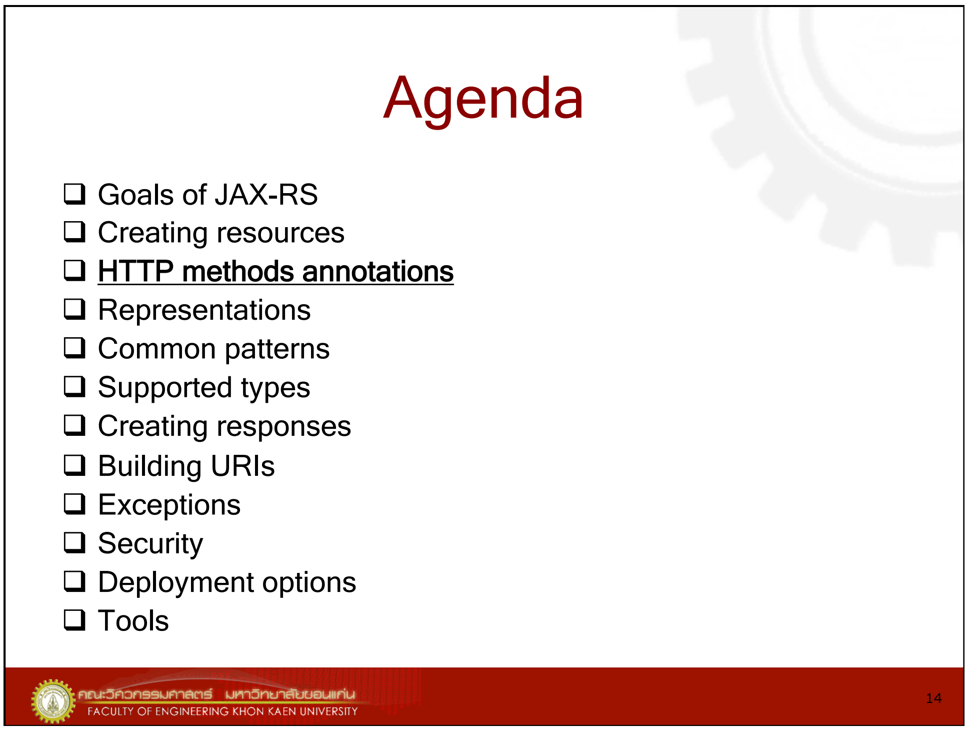
12



Example:@PathParam, @QueryParam

```
@Path("/items/")
@Consumes(“application/xml”)
public class ItemsResource {

  // Example request: http://host/catalog/items/?start=0
  @GET
  ItemsConverter get(@QueryParam("start")int start) {
     ...
  }

  // Example request: http://host/catalog/items/123
  @Path("{id}/")
  ItemResource getItemResource(@PathParam("id")Long id){
  ...
  }
```

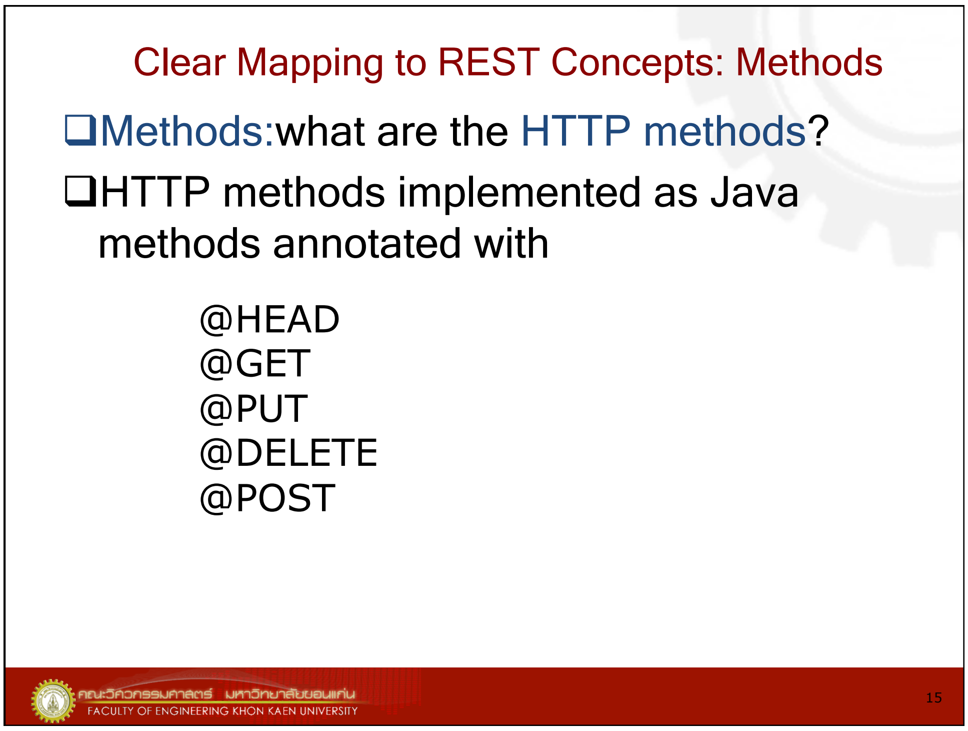
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 13



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ <u>**HTTP methods annotations**</u>
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 14



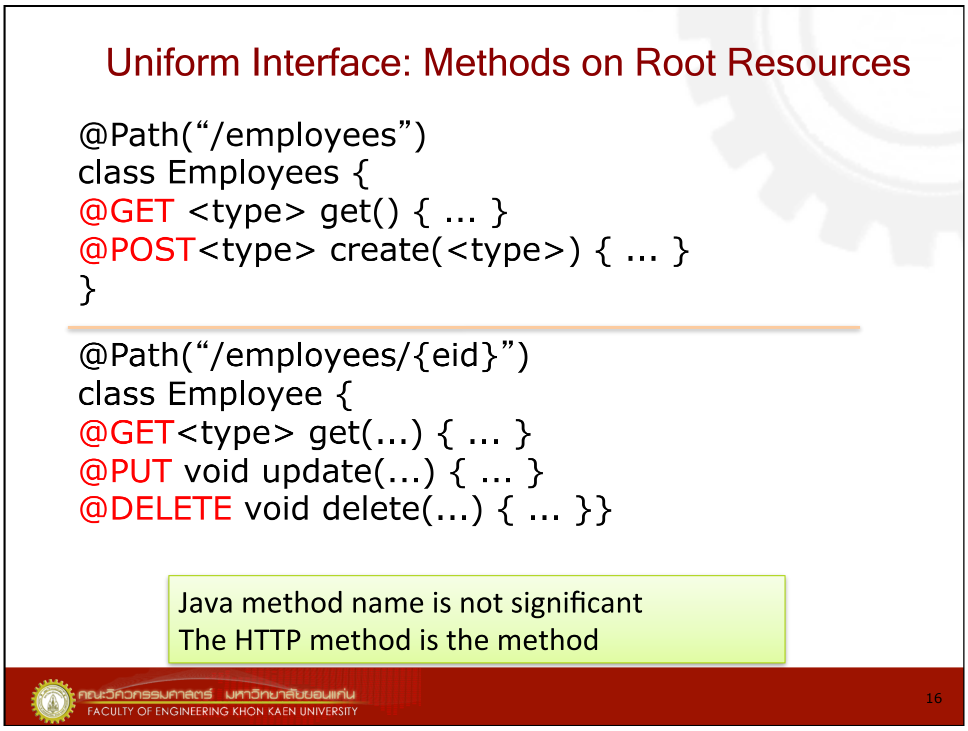
Clear Mapping to REST Concepts: Methods

- ❑Methods:what are the HTTP methods?
- ❑HTTP methods implemented as Java methods annotated with

```
@HEAD
@GET
@PUT
@DELETE
@POST
```

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

15



Uniform Interface: Methods on Root Resources

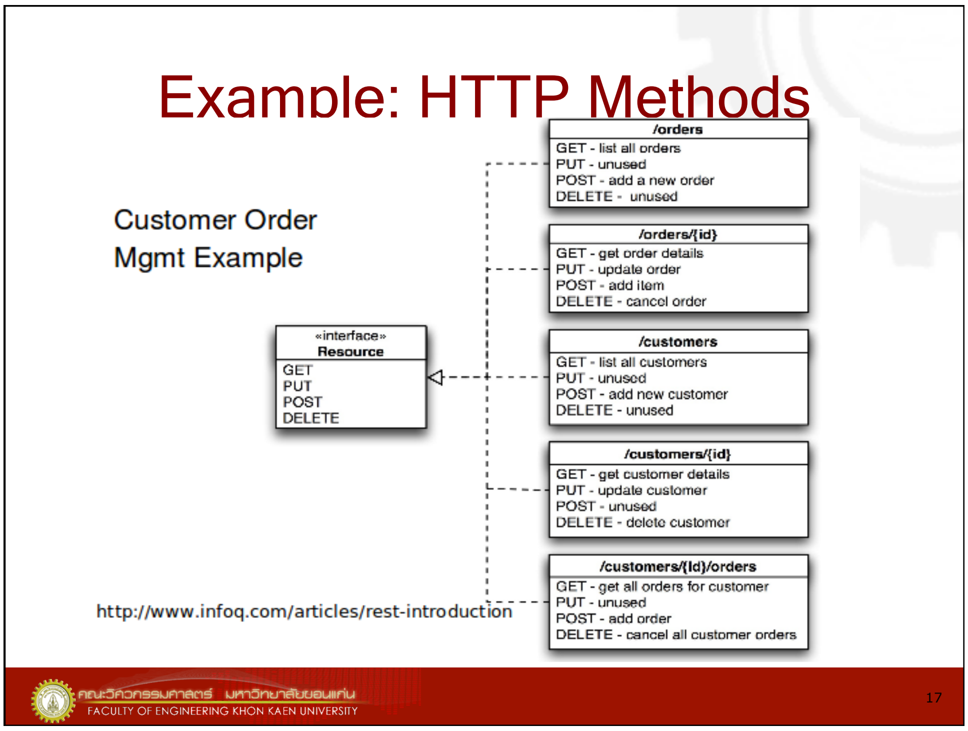
```
@Path(“/employees”)
class Employees {
@GET <type> get() { … }
@POST<type> create(<type>) { … }
}
```

```
@Path(“/employees/{eid}”)
class Employee {
@GET<type> get(...) { … }
@PUT void update(...) { … }
@DELETE void delete(...) { … }}
```

Java method name is not significant
The HTTP method is the method

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

16



Example: HTTP Methods

Customer Order
Mgmt Example

«interface»
Resource
GET
PUT
POST
DELETE

/orders
GET - list all orders
PUT - unused
POST - add a new order
DELETE - unused

/orders/{id}
GET - get order details
PUT - update order
POST - add item
DELETE - cancel order

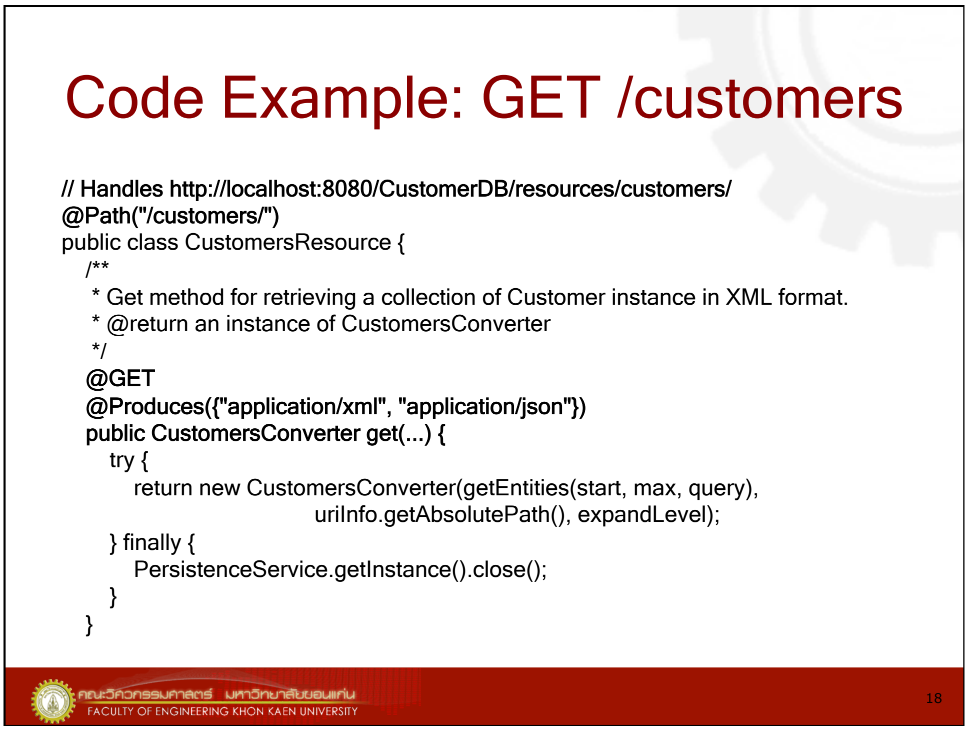
/customers
GET - list all customers
PUT - unused
POST - add new customer
DELETE - unused

/customers/{id}
GET - get customer details
PUT - update customer
POST - unused
DELETE - delete customer

/customers/{id}/orders
GET - get all orders for customer
PUT - unused
POST - add order
DELETE - cancel all customer orders

http://www.infoq.com/articles/rest-introduction

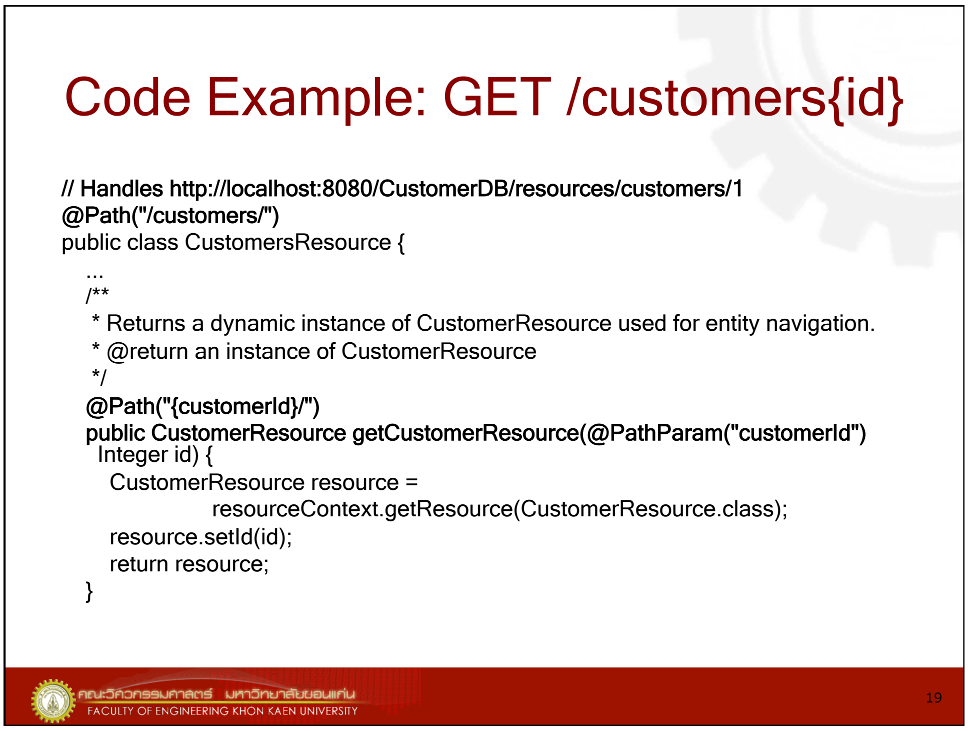
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 17



Code Example: GET /customers

```
// Handles http://localhost:8080/CustomerDB/resources/customers/
@Path("/customers/")
public class CustomersResource {
   /**
    * Get method for retrieving a collection of Customer instance in XML format.
    * @return an instance of CustomersConverter
    */
   @GET
   @Produces({"application/xml", "application/json"})
   public CustomersConverter get(...) {
      try {
         return new CustomersConverter(getEntities(start, max, query),
                              uriInfo.getAbsolutePath(), expandLevel);
      } finally {
         PersistenceService.getInstance().close();
      }
   }
```

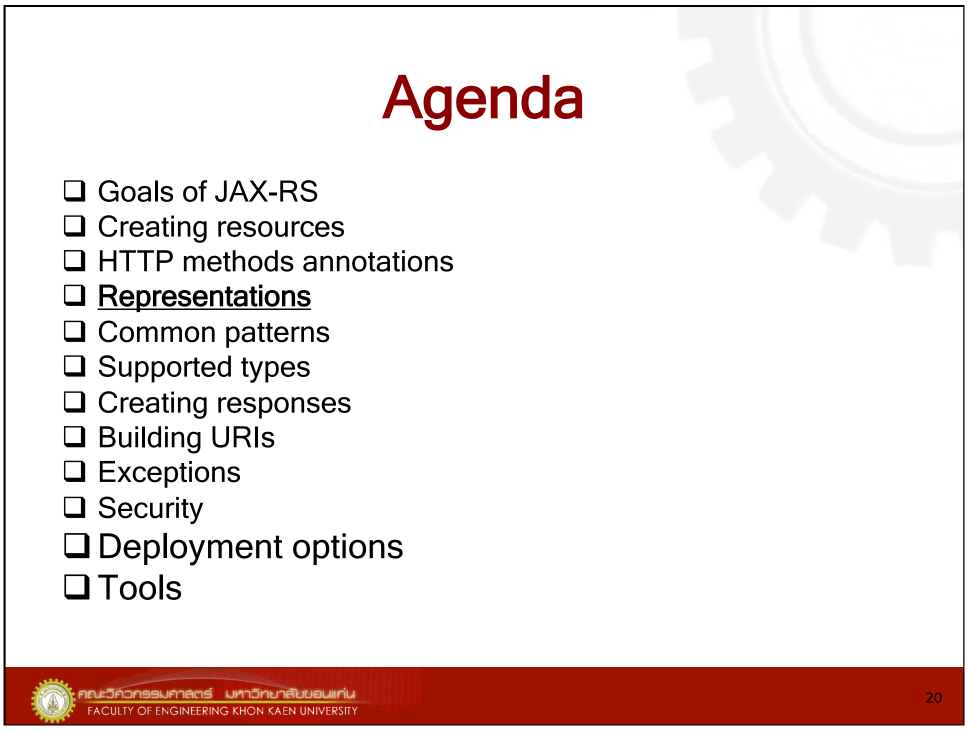
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 18



Code Example: GET /customers{id}

```
// Handles http://localhost:8080/CustomerDB/resources/customers/1
@Path("/customers/")
public class CustomersResource {
   ...
   /**
    * Returns a dynamic instance of CustomerResource used for entity navigation.
    * @return an instance of CustomerResource
    */
   @Path("{customerId}/")
   public CustomerResource getCustomerResource(@PathParam("customerId")
     Integer id) {
       CustomerResource resource =
                resourceContext.getResource(CustomerResource.class);
       resource.setId(id);
       return resource;
   }
```

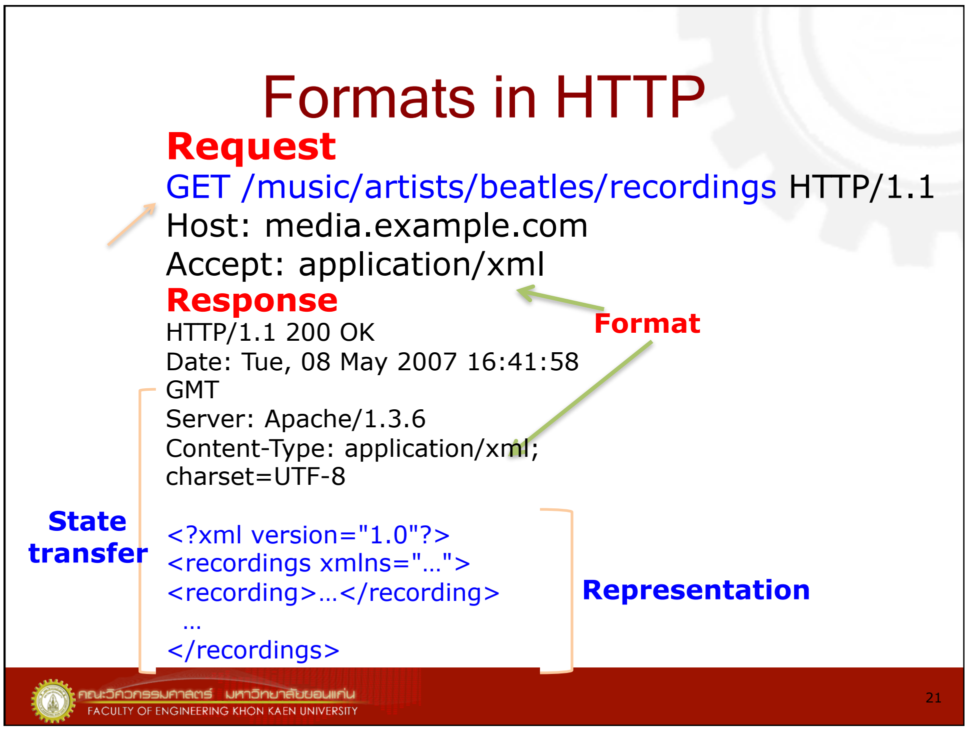
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 19



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 20



Formats in HTTP

Request

```
GET /music/artists/beatles/recordings HTTP/1.1
Host: media.example.com
Accept: application/xml
```

Response

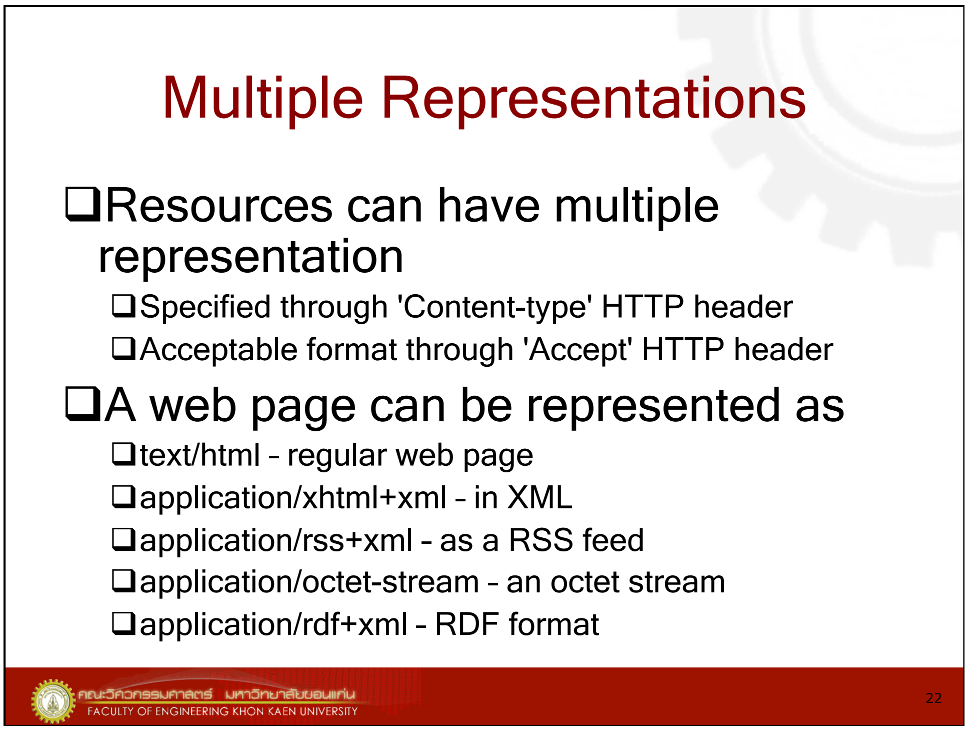
```
HTTP/1.1 200 OK
Date: Tue, 08 May 2007 16:41:58
GMT
Server: Apache/1.3.6
Content-Type: application/xml;
charset=UTF-8

<?xml version="1.0"?>
<recordings xmlns="...">
<recording>...</recording>
  ...
</recordings>
```

Format

State transfer

Representation

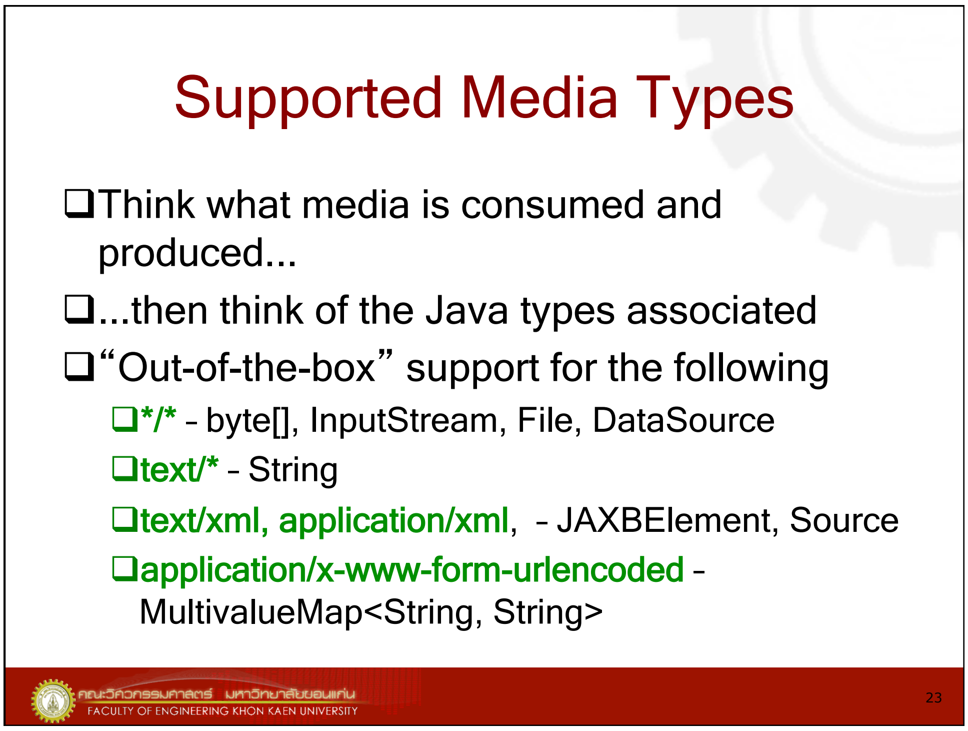
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 21



Multiple Representations

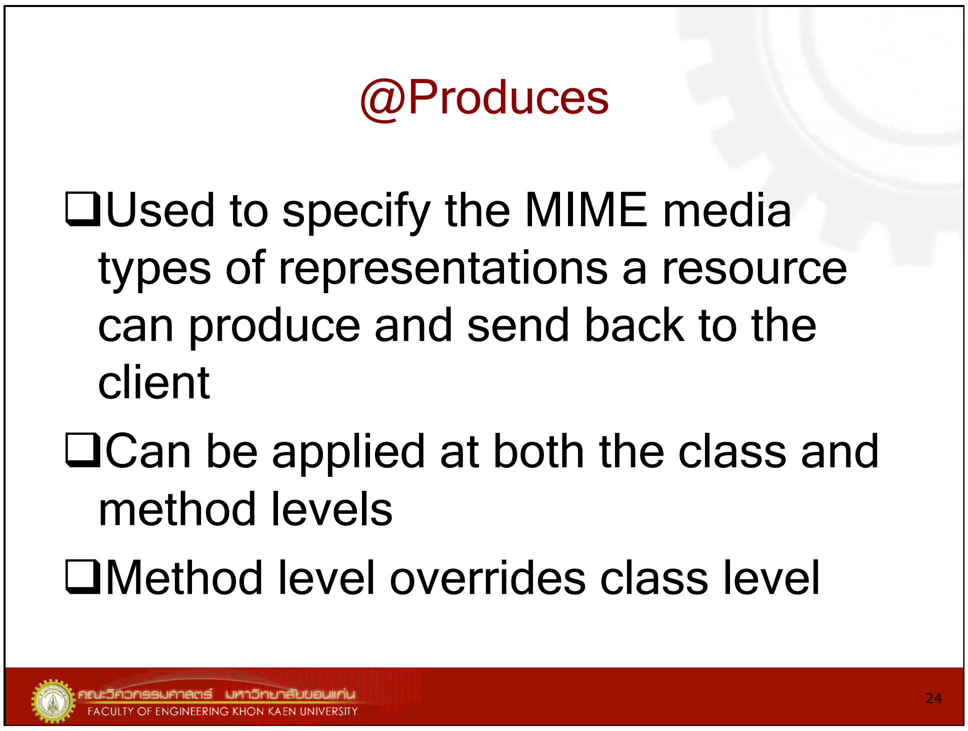
- ❑Resources can have multiple representation
  - ❑Specified through 'Content-type' HTTP header
  - ❑Acceptable format through 'Accept' HTTP header
- ❑A web page can be represented as
  - ❑text/html - regular web page
  - ❑application/xhtml+xml - in XML
  - ❑application/rss+xml - as a RSS feed
  - ❑application/octet-stream - an octet stream
  - ❑application/rdf+xml - RDF format

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 22



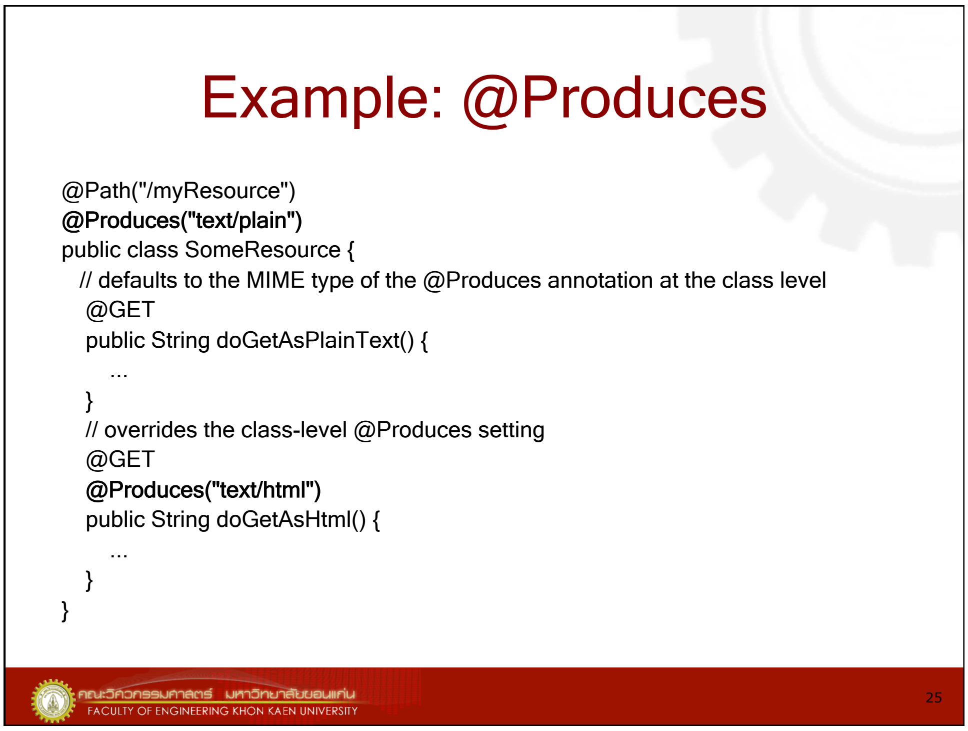
Supported Media Types

- ❑Think what media is consumed and produced...
- ❑...then think of the Java types associated
- ❑“Out-of-the-box” support for the following
  - ❑*/* - byte[], InputStream, File, DataSource
  - ❑text/* - String
  - ❑text/xml, application/xml, - JAXBElement, Source
  - ❑application/x-www-form-urlencoded - MultivalueMap<String, String>



@Produces

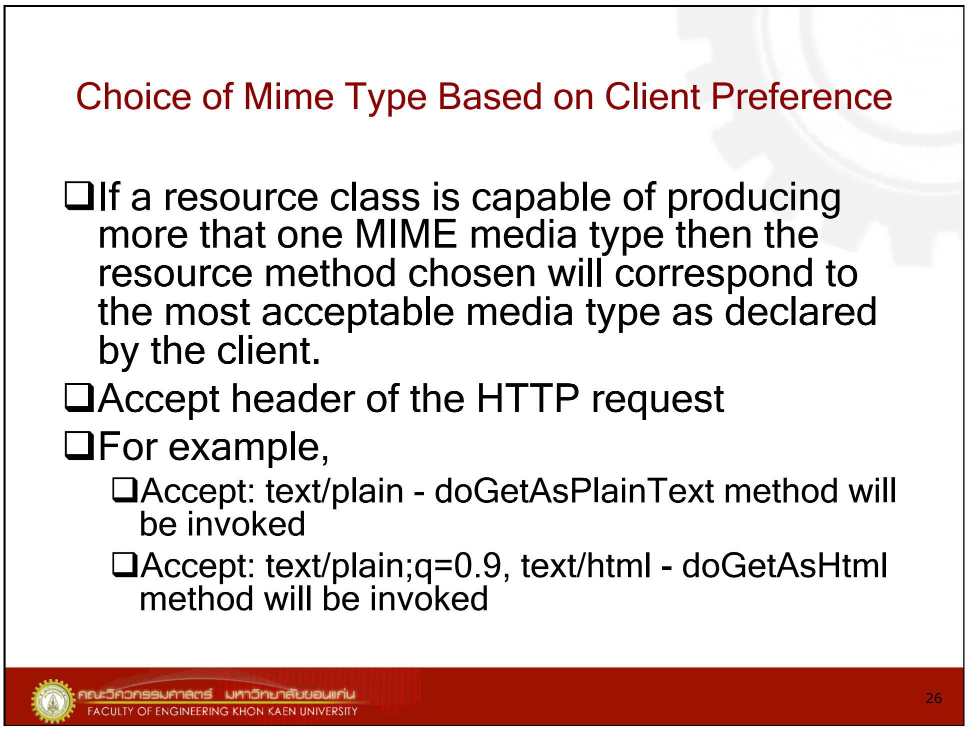
- ❑Used to specify the MIME media types of representations a resource can produce and send back to the client
- ❑Can be applied at both the class and method levels
- ❑Method level overrides class level



Example: @Produces

```
@Path("/myResource")
@Produces("text/plain")
public class SomeResource {
  // defaults to the MIME type of the @Produces annotation at the class level
    @GET
    public String doGetAsPlainText() {
        ...
    }
    // overrides the class-level @Produces setting
    @GET
    @Produces("text/html")
    public String doGetAsHtml() {
        ...
    }
}
```

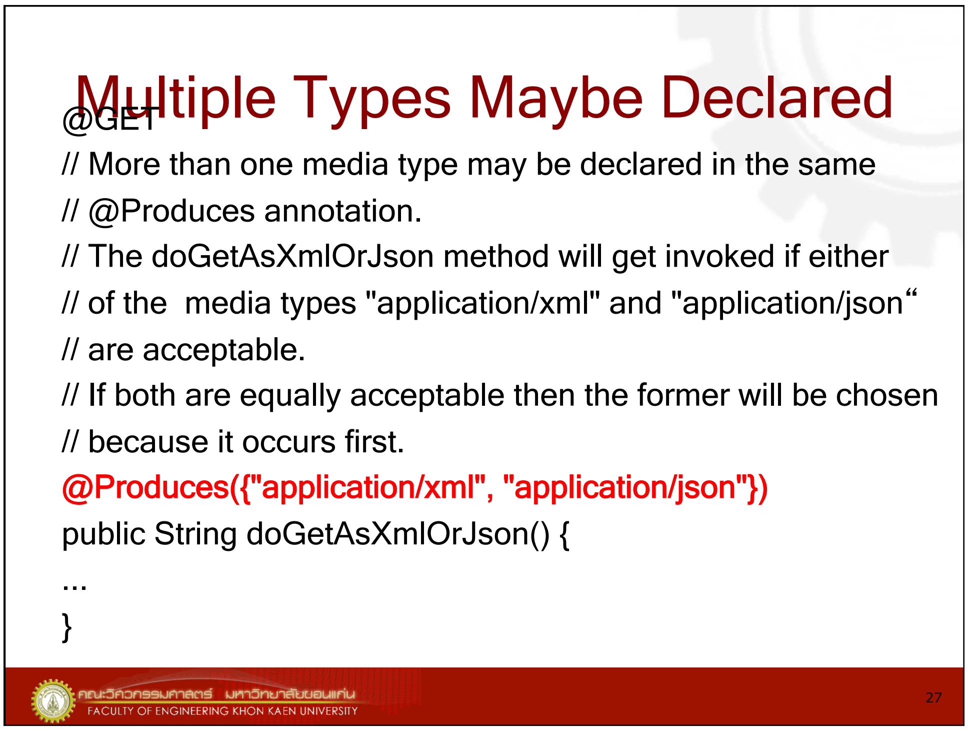
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 25



Choice of Mime Type Based on Client Preference

- If a resource class is capable of producing more that one MIME media type then the resource method chosen will correspond to the most acceptable media type as declared by the client.
- Accept header of the HTTP request
- For example,
  - Accept: text/plain - doGetAsPlainText method will be invoked
  - Accept: text/plain;q=0.9, text/html - doGetAsHtml method will be invoked

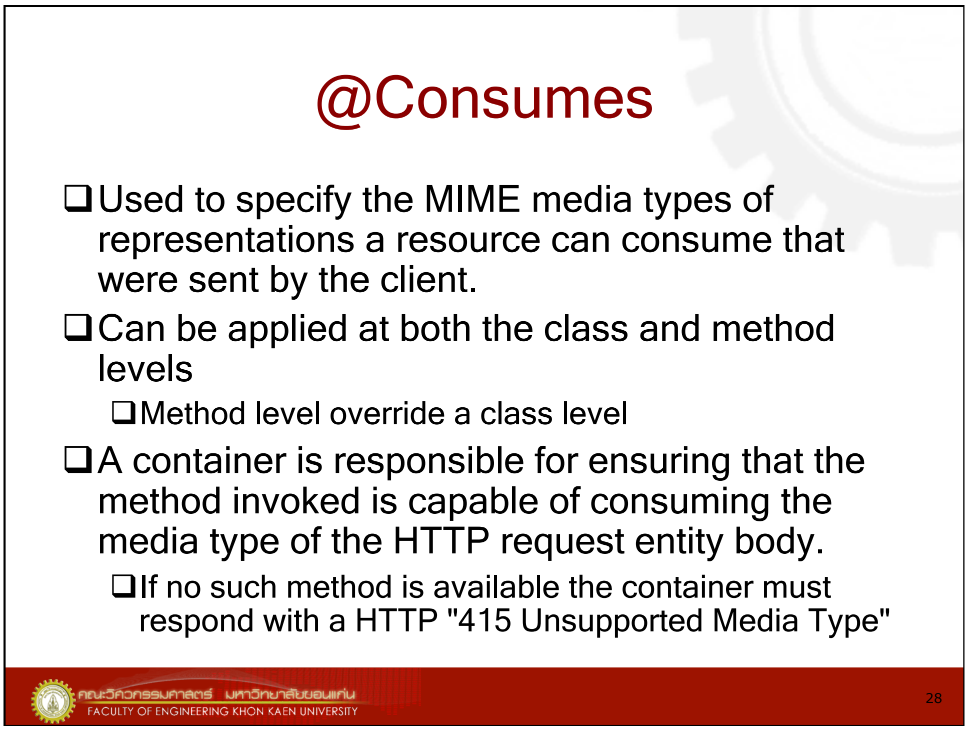
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 26



Multiple Types Maybe Declared

```
@GET
// More than one media type may be declared in the same
// @Produces annotation.
// The doGetAsXmlOrJson method will get invoked if either
// of the  media types "application/xml" and "application/json“
// are acceptable.
// If both are equally acceptable then the former will be chosen
// because it occurs first.
@Produces({"application/xml", "application/json"})
public String doGetAsXmlOrJson() {
...
}
```

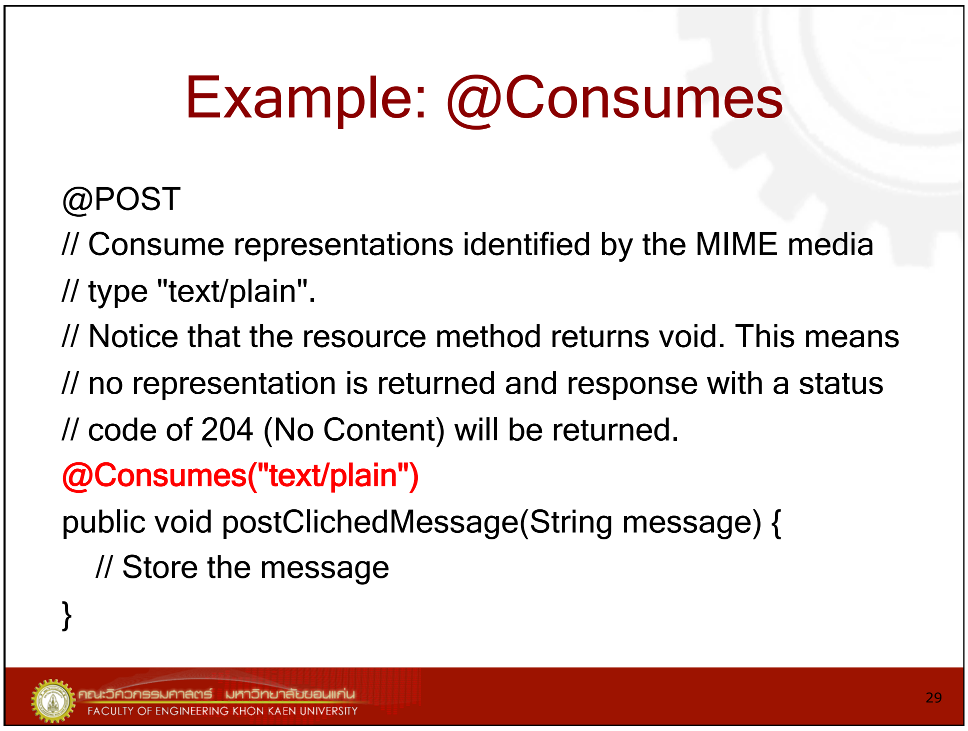
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 27



@Consumes

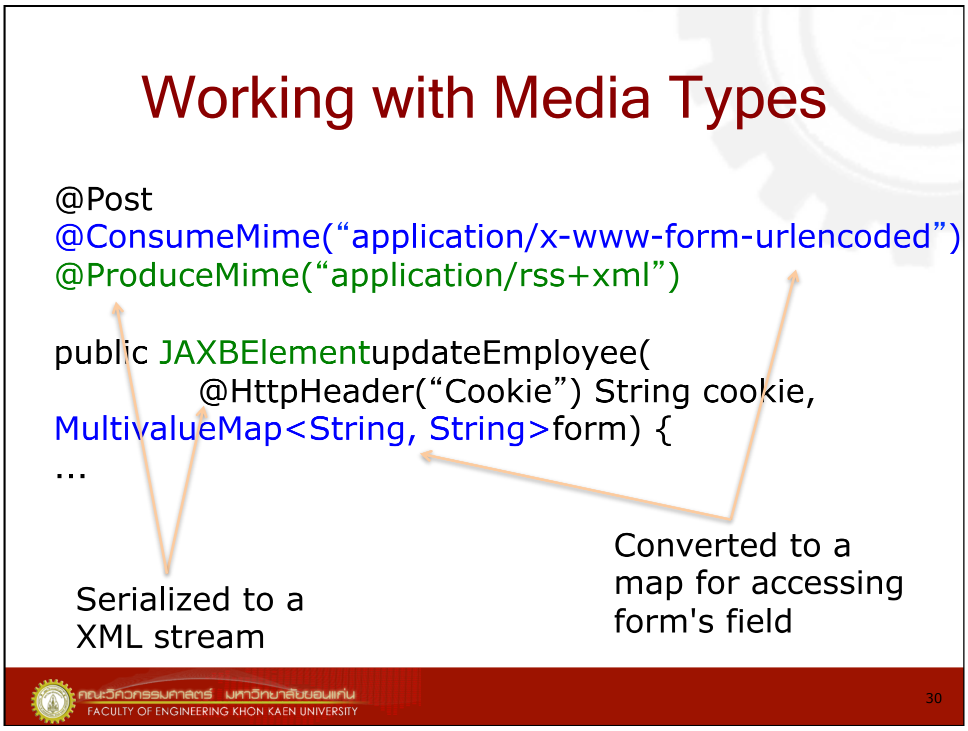
- ❑ Used to specify the MIME media types of representations a resource can consume that were sent by the client.
- ❑ Can be applied at both the class and method levels
  - ❑ Method level override a class level
- ❑ A container is responsible for ensuring that the method invoked is capable of consuming the media type of the HTTP request entity body.
  - ❑ If no such method is available the container must respond with a HTTP "415 Unsupported Media Type"

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 28



Example: @Consumes

```
@POST
// Consume representations identified by the MIME media
// type "text/plain".
// Notice that the resource method returns void. This means
// no representation is returned and response with a status
// code of 204 (No Content) will be returned.
@Consumes("text/plain")
public void postClichedMessage(String message) {
   // Store the message
}
```

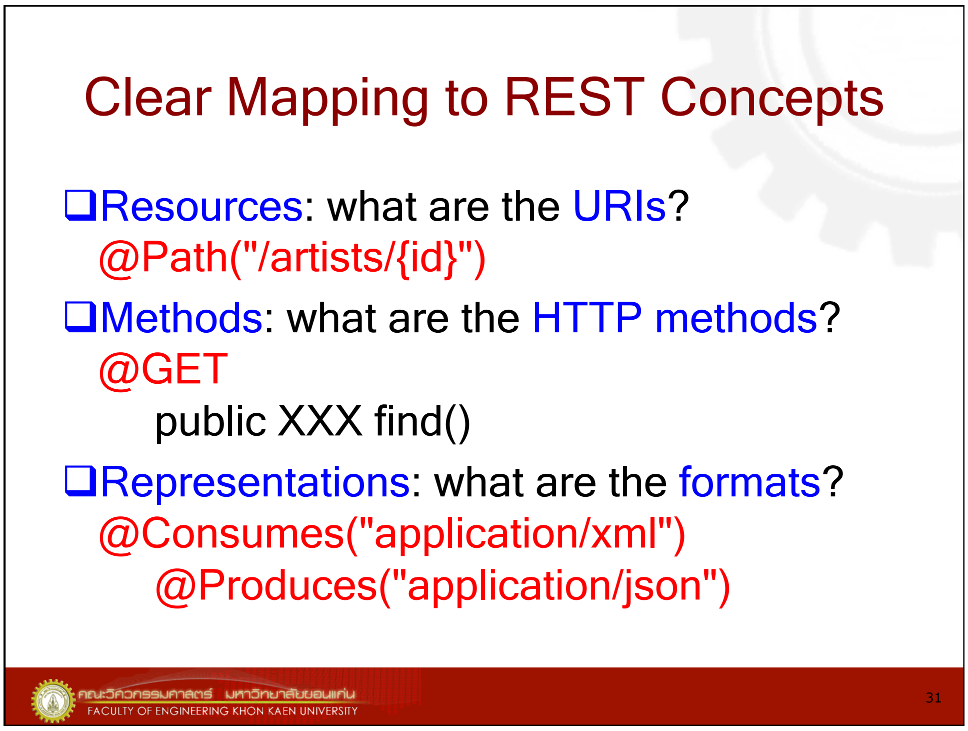


Working with Media Types

```
@Post
@ConsumeMime("application/x-www-form-urlencoded")
@ProduceMime("application/rss+xml")

public JAXBElementupdateEmployee(
           @HttpHeader("Cookie") String cookie,
MultivalueMap<String, String>form) {
...
```

Serialized to a XML stream

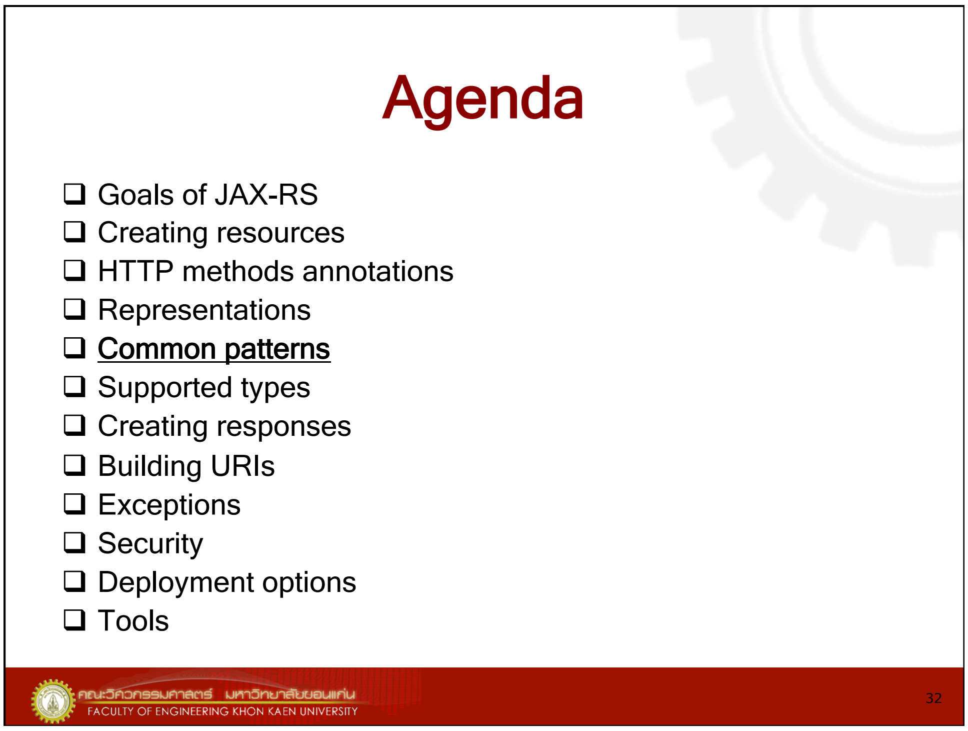
Converted to a map for accessing form's field



Clear Mapping to REST Concepts

- Resources: what are the URIs?
  @Path("/artists/{id}")
- Methods: what are the HTTP methods?
  @GET
  public XXX find()
- Representations: what are the formats?
  @Consumes("application/xml")
  @Produces("application/json")

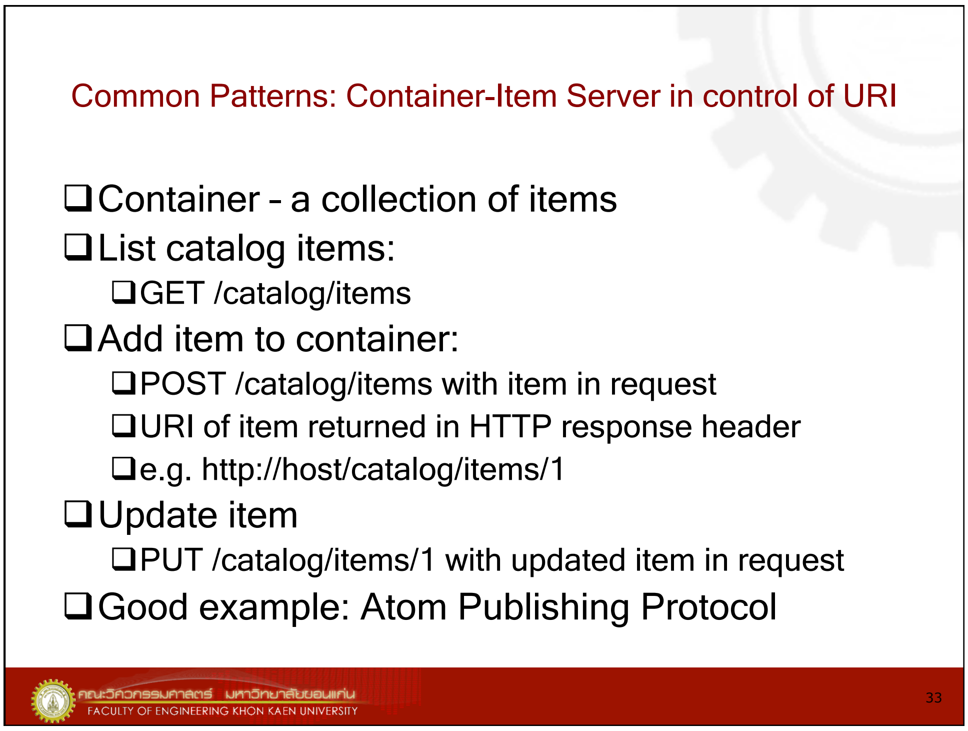
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 31



Agenda

- Goals of JAX-RS
- Creating resources
- HTTP methods annotations
- Representations
- **Common patterns**
- Supported types
- Creating responses
- Building URIs
- Exceptions
- Security
- Deployment options
- Tools

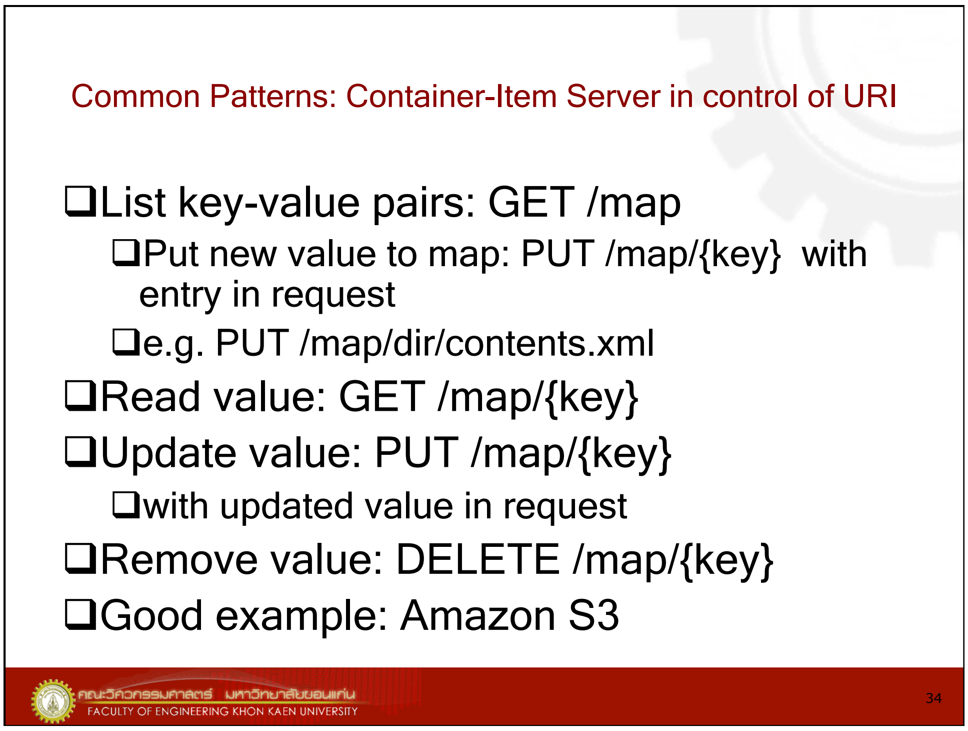
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 32



Common Patterns: Container-Item Server in control of URI

- ❑Container - a collection of items
- ❑List catalog items:
  - ❑GET /catalog/items
- ❑Add item to container:
  - ❑POST /catalog/items with item in request
  - ❑URI of item returned in HTTP response header
  - ❑e.g. http://host/catalog/items/1
- ❑Update item
  - ❑PUT /catalog/items/1 with updated item in request
- ❑Good example: Atom Publishing Protocol

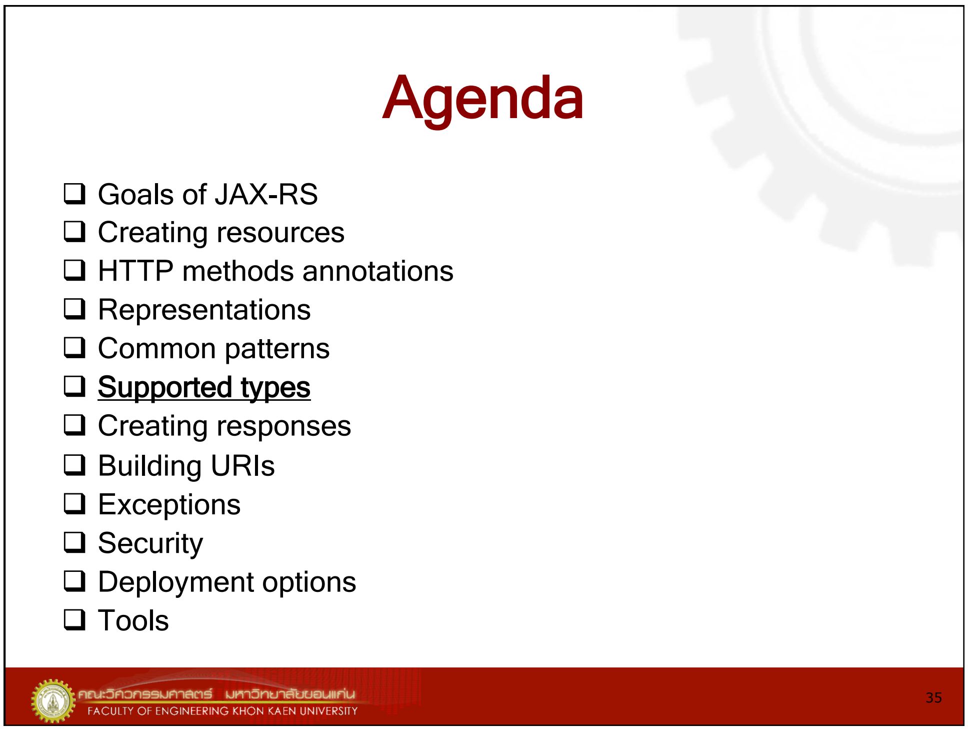
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 33



Common Patterns: Container-Item Server in control of URI

- ❑List key-value pairs: GET /map
  - ❑Put new value to map: PUT /map/{key} with entry in request
  - ❑e.g. PUT /map/dir/contents.xml
- ❑Read value: GET /map/{key}
- ❑Update value: PUT /map/{key}
  - ❑with updated value in request
- ❑Remove value: DELETE /map/{key}
- ❑Good example: Amazon S3

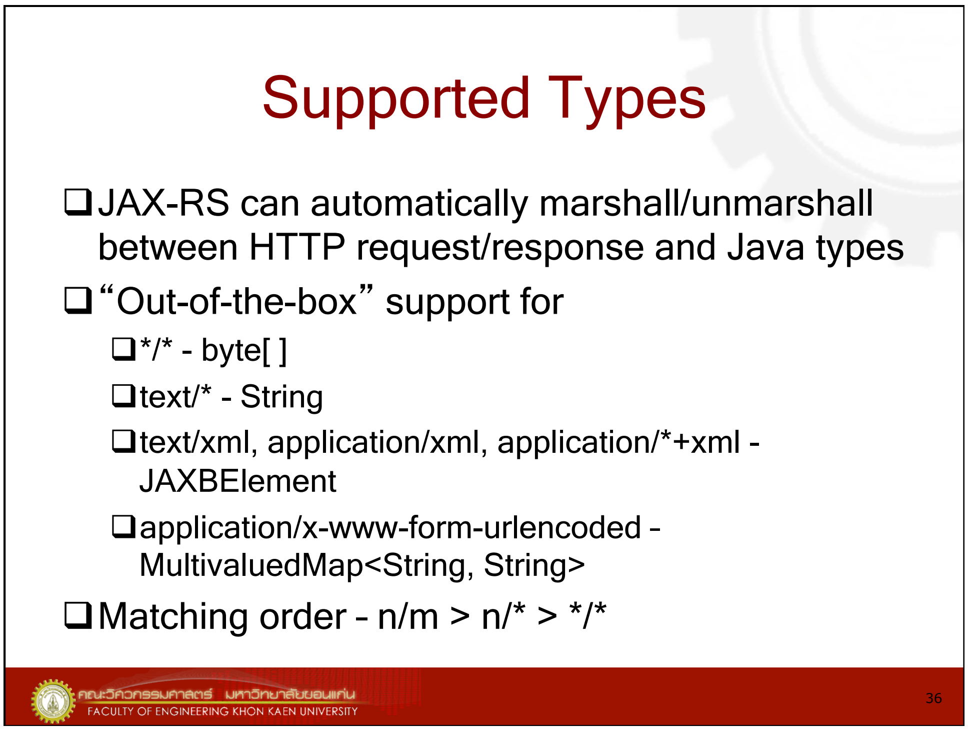
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 34



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ <u>Supported types</u>
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools

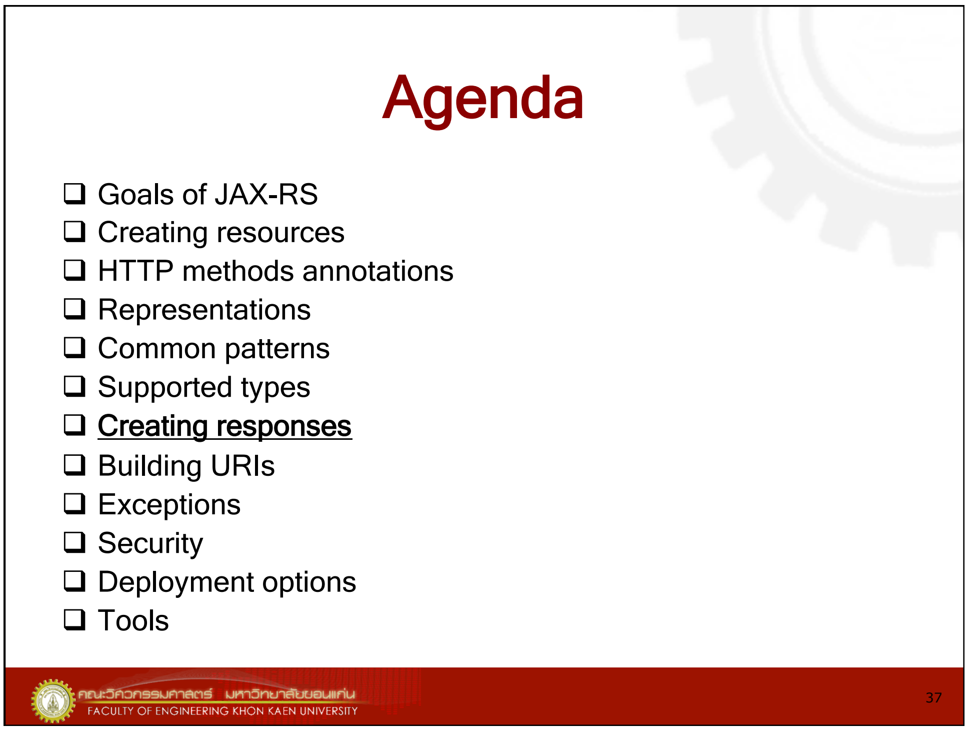
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 35



Supported Types

- ❑ JAX-RS can automatically marshall/unmarshall between HTTP request/response and Java types
- ❑ "Out-of-the-box" support for
  - ❑ */* - byte[ ]
  - ❑ text/* - String
  - ❑ text/xml, application/xml, application/*+xml - JAXBElement
  - ❑ application/x-www-form-urlencoded - MultivaluedMap<String, String>
- ❑ Matching order - n/m > n/* > */*

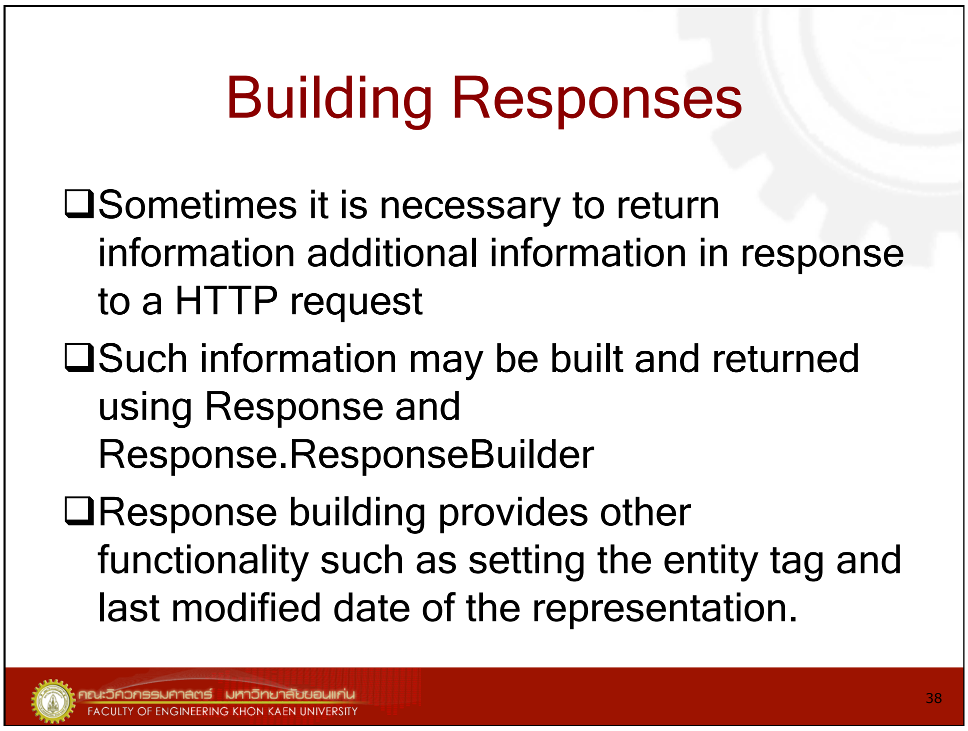
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 36



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools

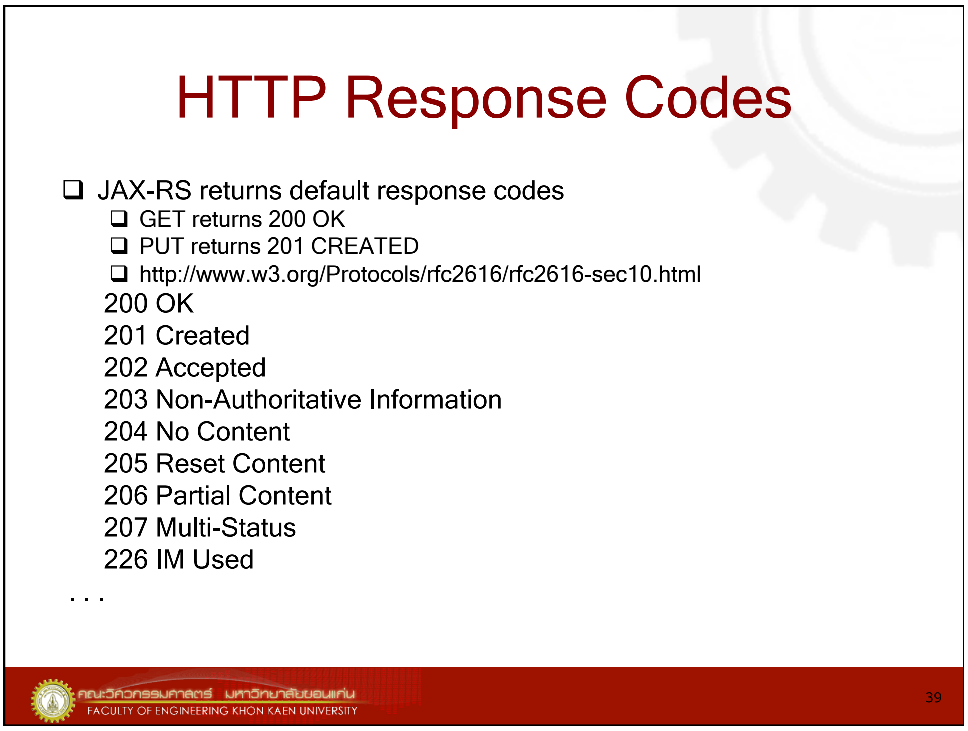
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 37



Building Responses

- ❑Sometimes it is necessary to return information additional information in response to a HTTP request
- ❑Such information may be built and returned using Response and Response.ResponseBuilder
- ❑Response building provides other functionality such as setting the entity tag and last modified date of the representation.

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 38

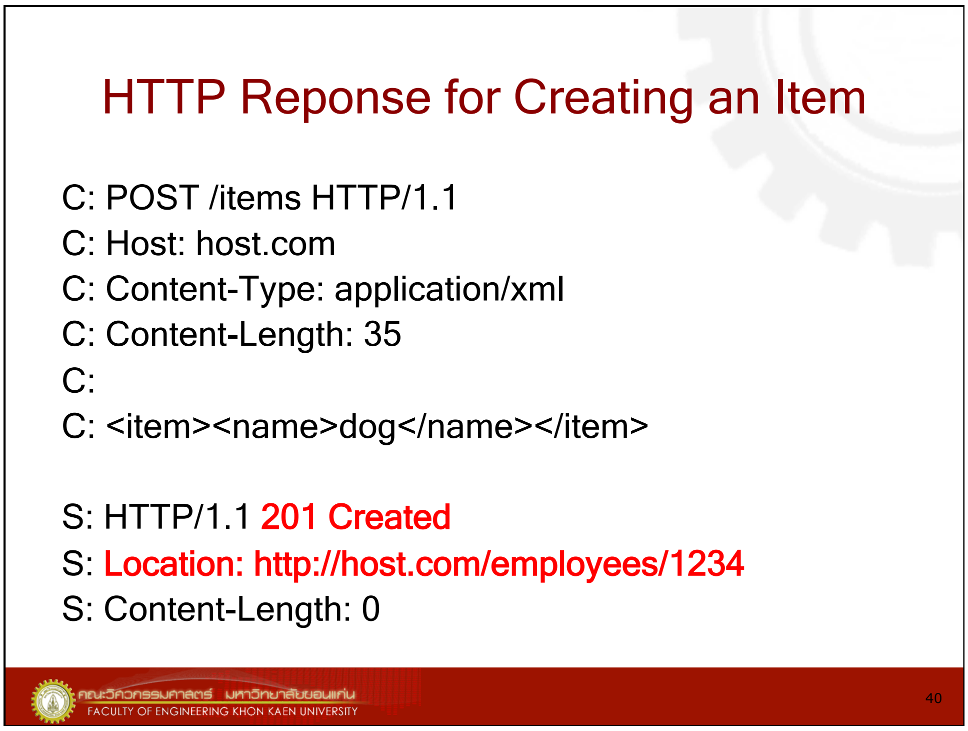


HTTP Response Codes

- JAX-RS returns default response codes
  - GET returns 200 OK
  - PUT returns 201 CREATED
  - http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

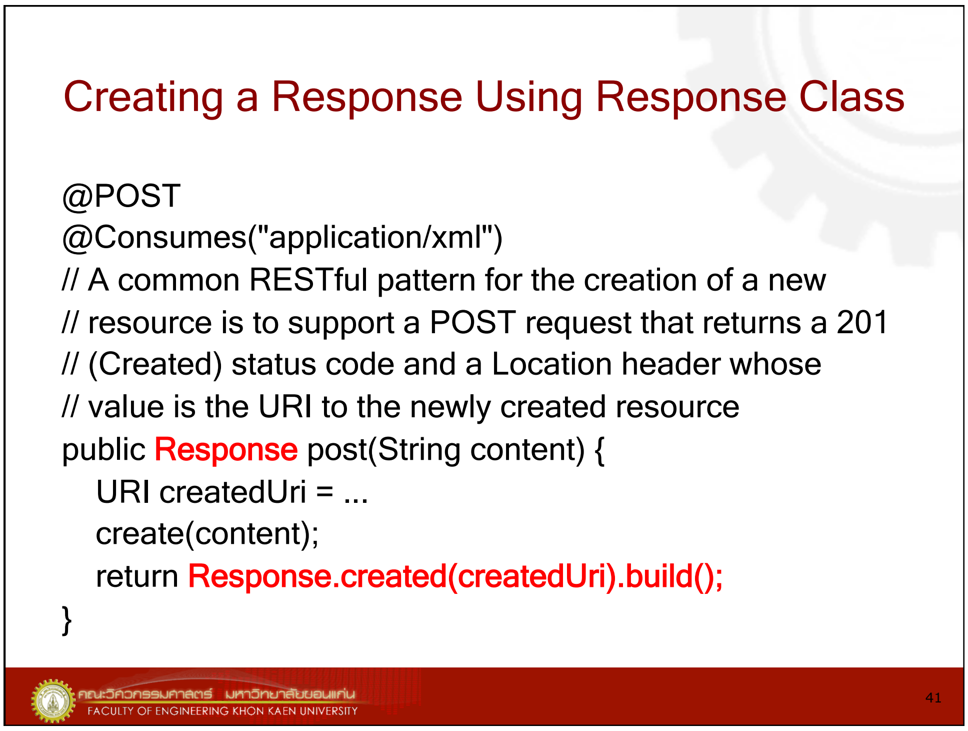
  200 OK
  201 Created
  202 Accepted
  203 Non-Authoritative Information
  204 No Content
  205 Reset Content
  206 Partial Content
  207 Multi-Status
  226 IM Used

. . .



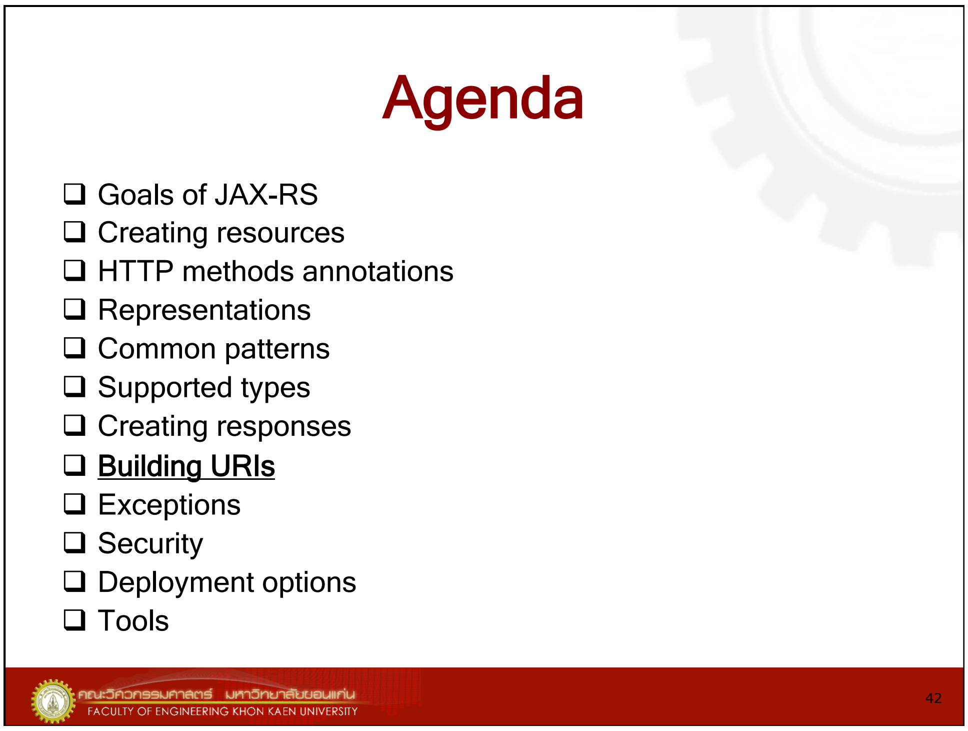
HTTP Reponse for Creating an Item

```
C: POST /items HTTP/1.1
C: Host: host.com
C: Content-Type: application/xml
C: Content-Length: 35
C:
C: <item><name>dog</name></item>

S: HTTP/1.1 201 Created
S: Location: http://host.com/employees/1234
S: Content-Length: 0
```



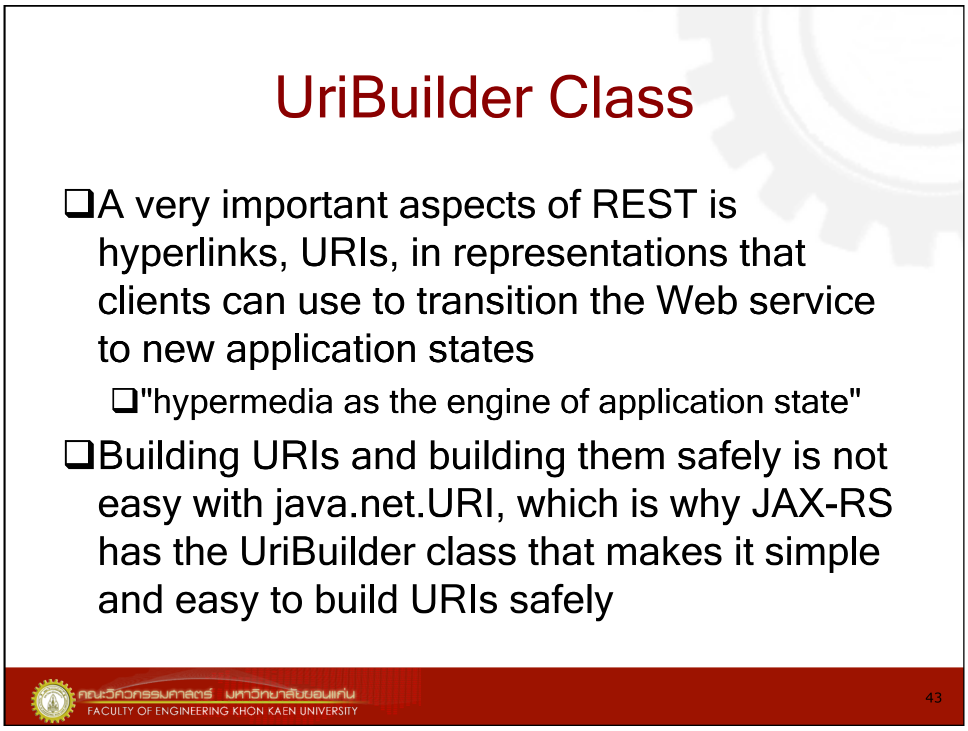
Creating a Response Using Response Class

```
@POST
@Consumes("application/xml")
// A common RESTful pattern for the creation of a new
// resource is to support a POST request that returns a 201
// (Created) status code and a Location header whose
// value is the URI to the newly created resource
public Response post(String content) {
   URI createdUri = ...
   create(content);
   return Response.created(createdUri).build();
}
```



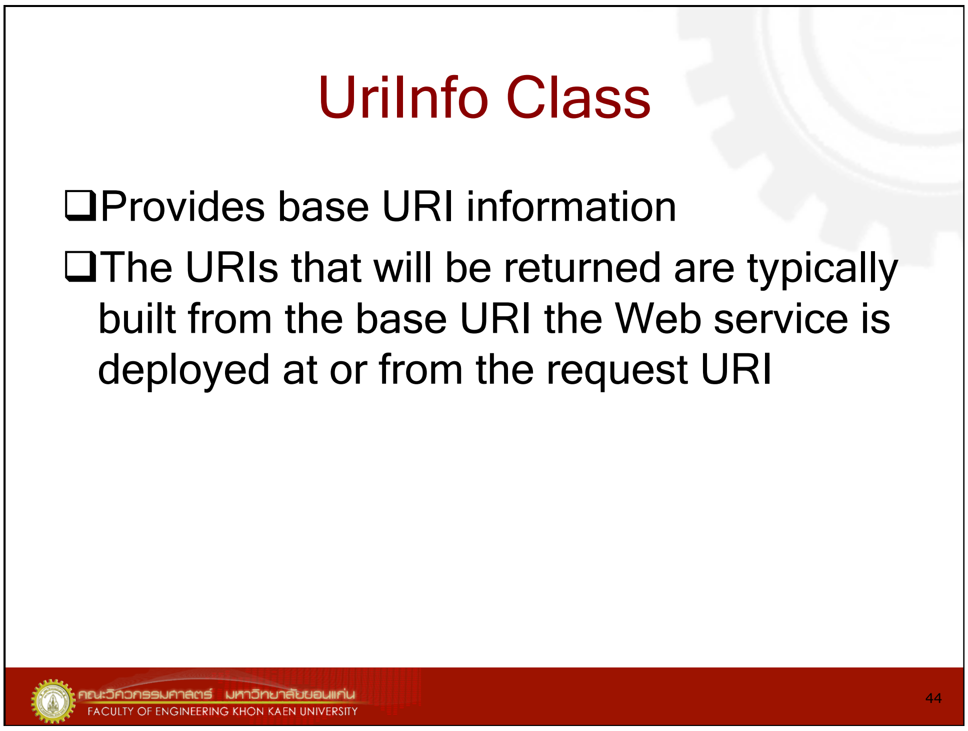
Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ **Building URIs**
- ❑ Exceptions
- ❑ Security
- ❑ Deployment options
- ❑ Tools



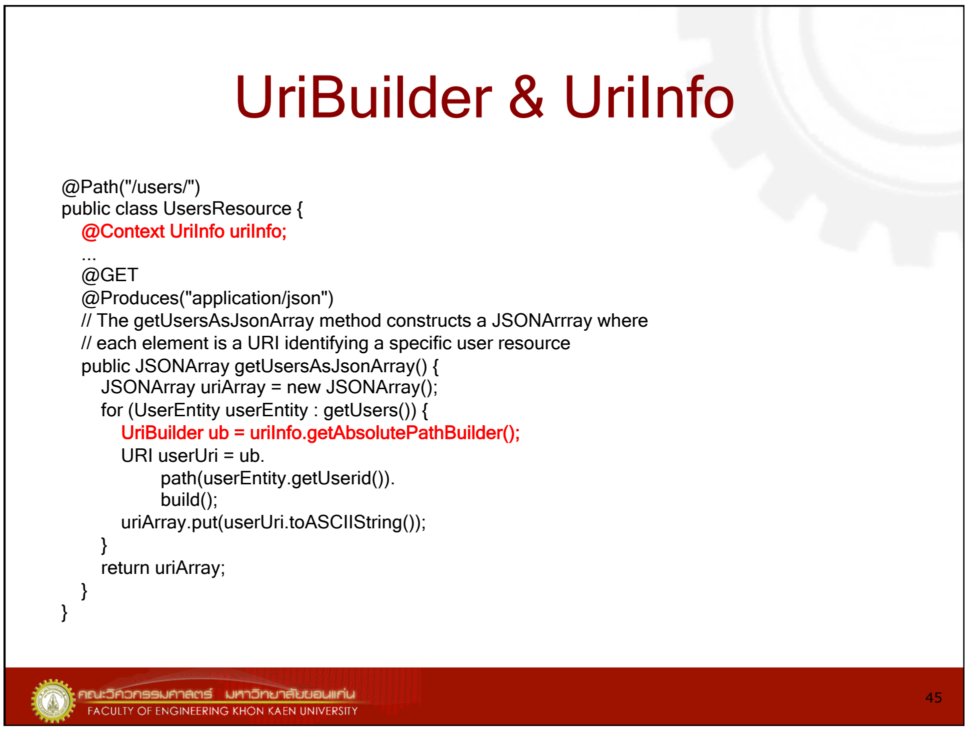
UriBuilder Class

- A very important aspects of REST is hyperlinks, URIs, in representations that clients can use to transition the Web service to new application states
  - "hypermedia as the engine of application state"
- Building URIs and building them safely is not easy with java.net.URI, which is why JAX-RS has the UriBuilder class that makes it simple and easy to build URIs safely



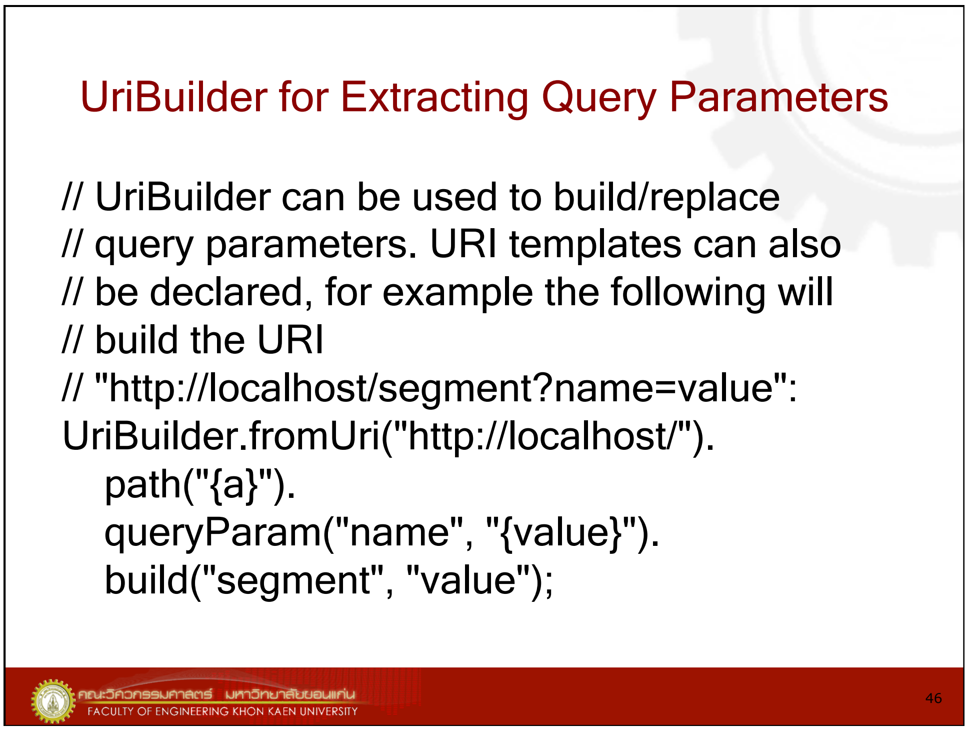
UriInfo Class

- Provides base URI information
- The URIs that will be returned are typically built from the base URI the Web service is deployed at or from the request URI



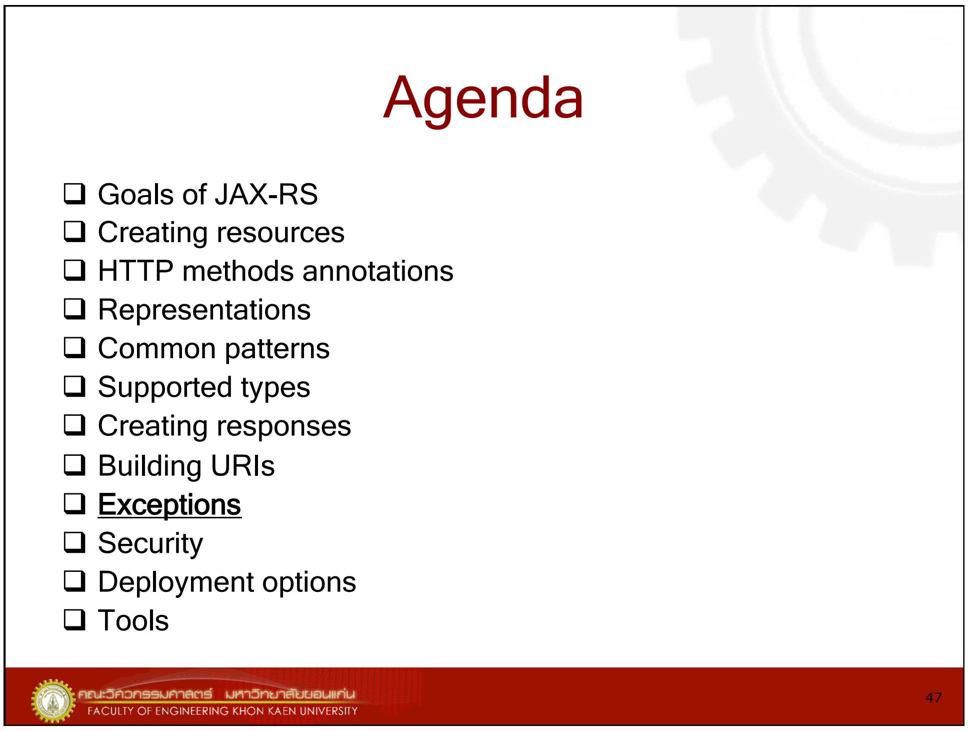
UriBuilder & UriInfo

```
@Path("/users/")
public class UsersResource {
   @Context UriInfo uriInfo;
   ...
   @GET
   @Produces("application/json")
   // The getUsersAsJsonArray method constructs a JSONArrray where
   // each element is a URI identifying a specific user resource
   public JSONArray getUsersAsJsonArray() {
      JSONArray uriArray = new JSONArray();
      for (UserEntity userEntity : getUsers()) {
         UriBuilder ub = uriInfo.getAbsolutePathBuilder();
         URI userUri = ub.
               path(userEntity.getUserid()).
               build();
         uriArray.put(userUri.toASCIIString());
      }
      return uriArray;
   }
}
```



UriBuilder for Extracting Query Parameters

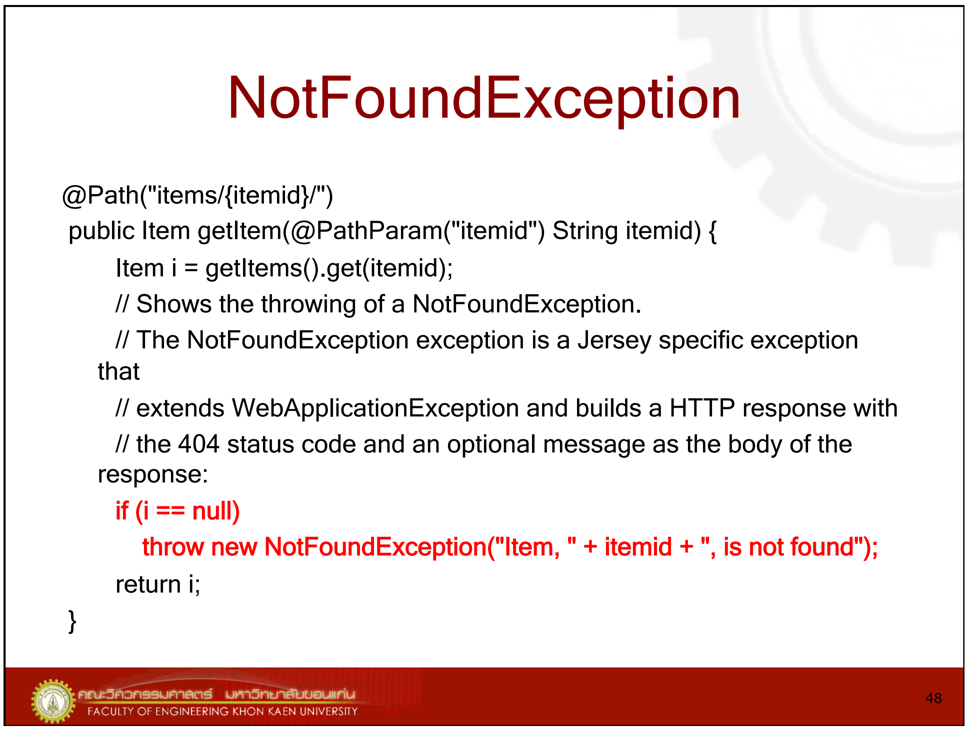
```
// UriBuilder can be used to build/replace
// query parameters. URI templates can also
// be declared, for example the following will
// build the URI
// "http://localhost/segment?name=value":
UriBuilder.fromUri("http://localhost/").
   path("{a}").
   queryParam("name", "{value}").
   build("segment", "value");
```



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ **Exceptions**
- ❑ Security
- ❑ Deployment options
- ❑ Tools

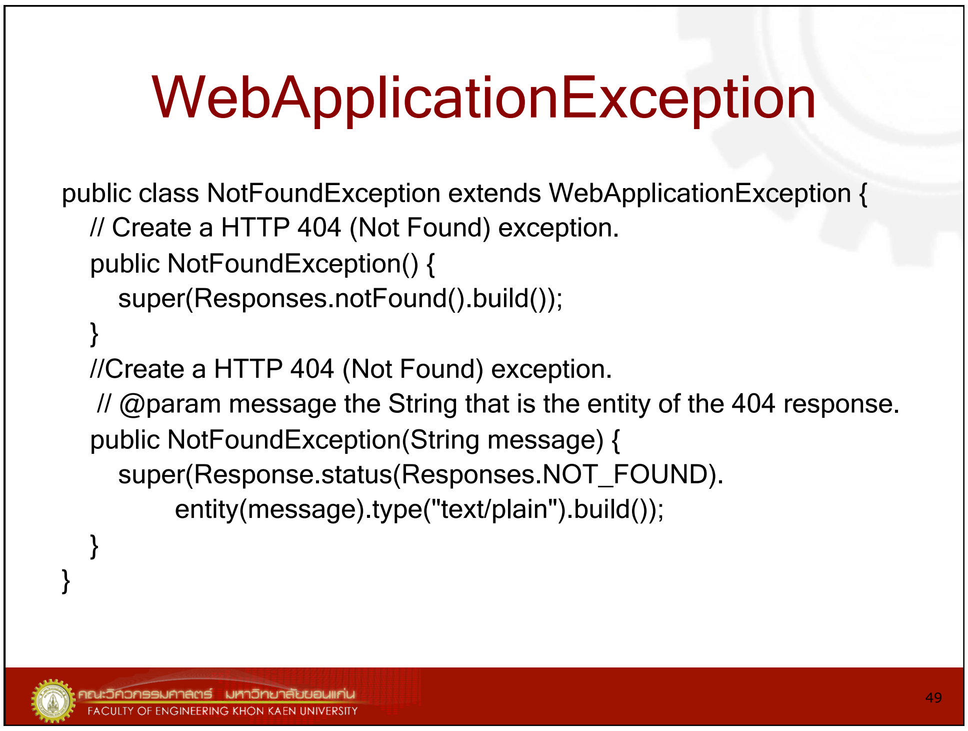
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 47



NotFoundException

```
@Path("items/{itemid}/")
 public Item getItem(@PathParam("itemid") String itemid) {
      Item i = getItems().get(itemid);
      // Shows the throwing of a NotFoundException.
      // The NotFoundException exception is a Jersey specific exception
    that
      // extends WebApplicationException and builds a HTTP response with
      // the 404 status code and an optional message as the body of the
    response:
      if (i == null)
          throw new NotFoundException("Item, " + itemid + ", is not found");
      return i;
 }
```

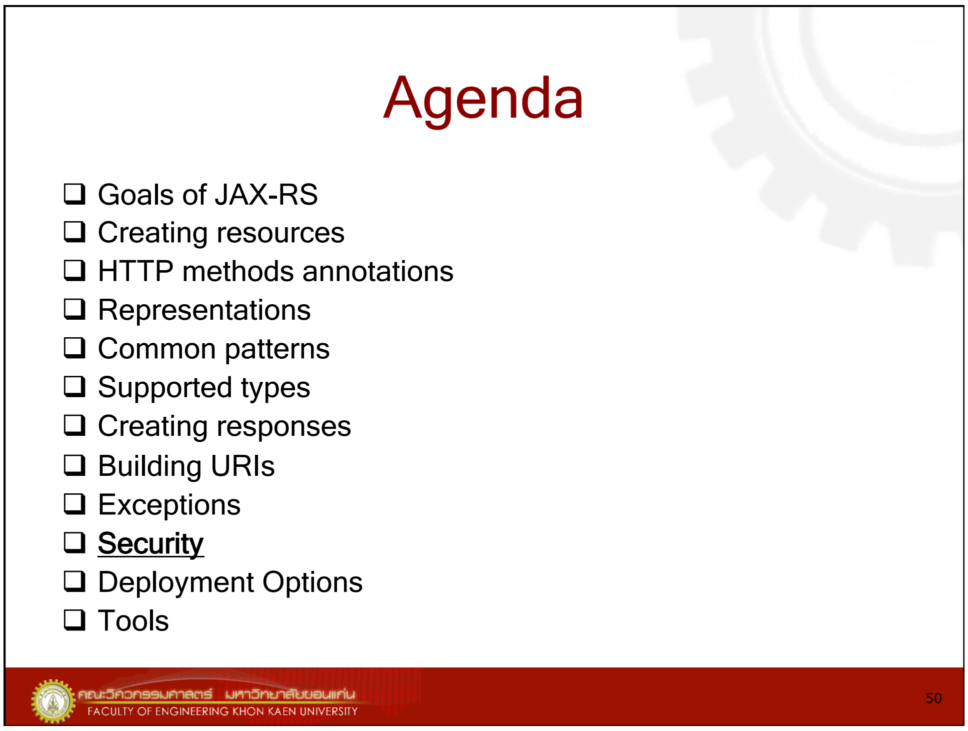
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 48



WebApplicationException

```
public class NotFoundException extends WebApplicationException {
    // Create a HTTP 404 (Not Found) exception.
    public NotFoundException() {
        super(Responses.notFound().build());
    }
    //Create a HTTP 404 (Not Found) exception.
     // @param message the String that is the entity of the 404 response.
    public NotFoundException(String message) {
        super(Response.status(Responses.NOT_FOUND).
                entity(message).type("text/plain").build());
    }
}
```

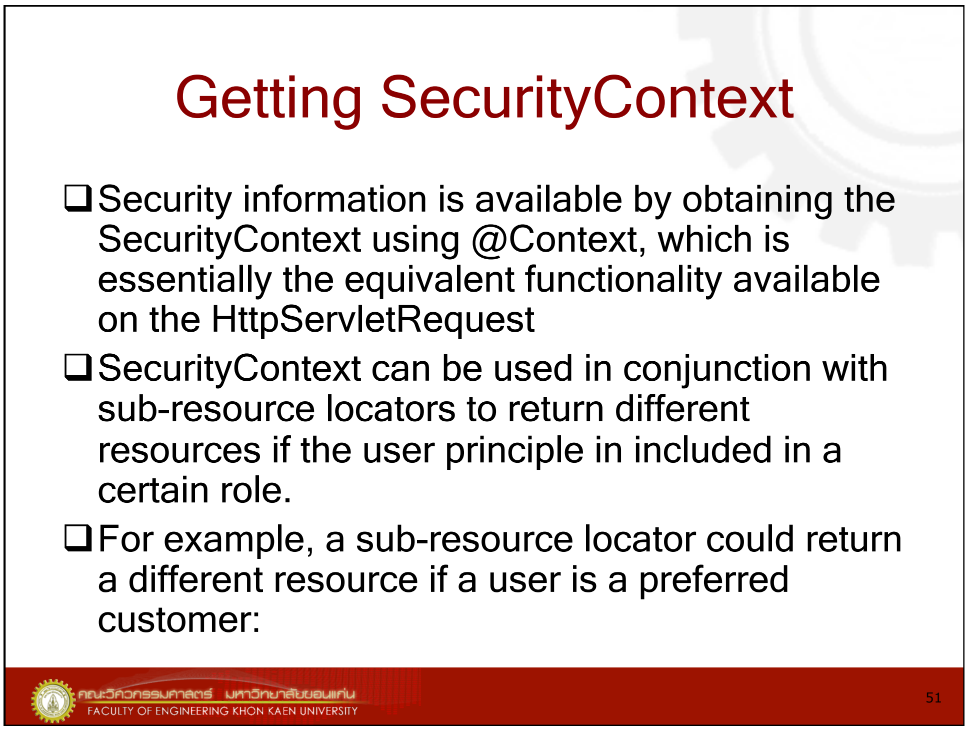
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 49



Agenda

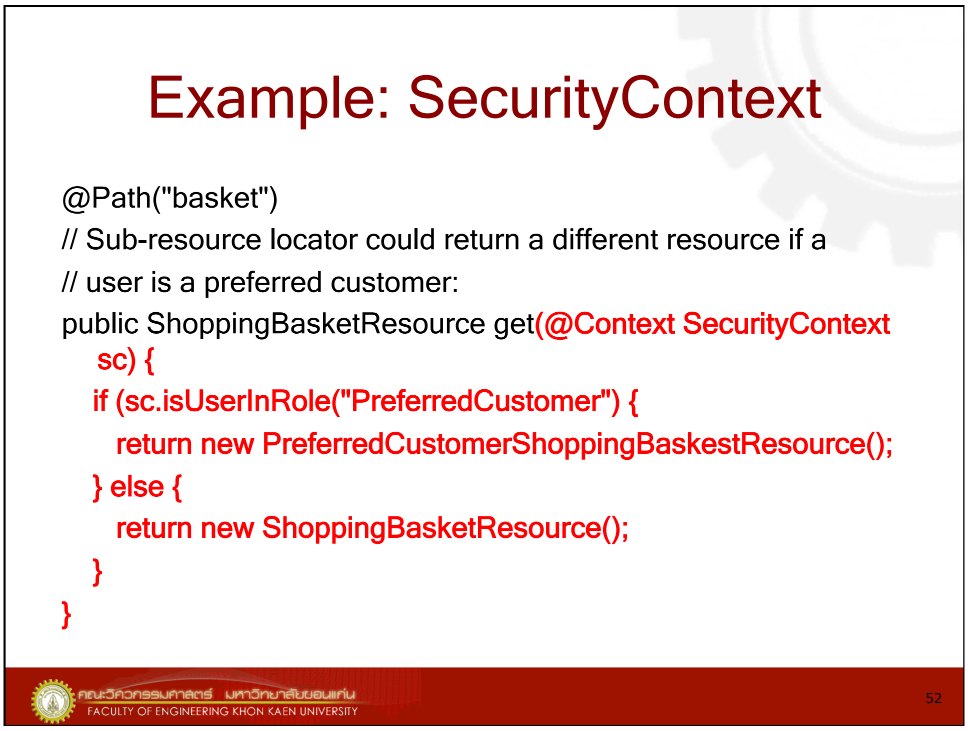
- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ **Security**
- ❑ Deployment Options
- ❑ Tools

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 50



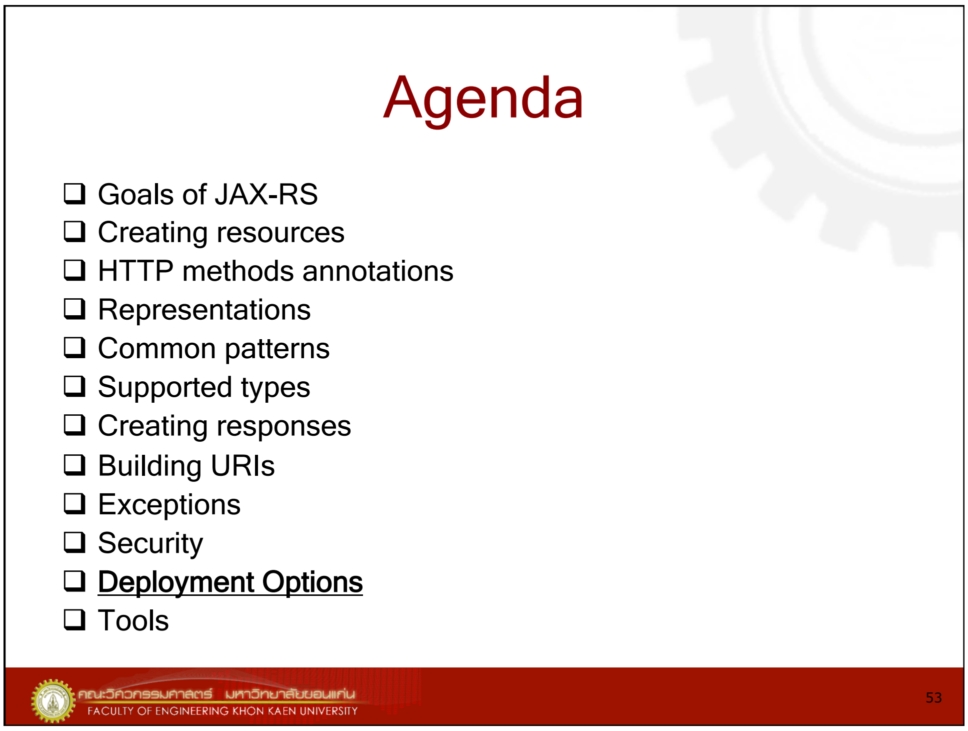
Getting SecurityContext

- Security information is available by obtaining the SecurityContext using @Context, which is essentially the equivalent functionality available on the HttpServletRequest
- SecurityContext can be used in conjunction with sub-resource locators to return different resources if the user principle in included in a certain role.
- For example, a sub-resource locator could return a different resource if a user is a preferred customer:



Example: SecurityContext

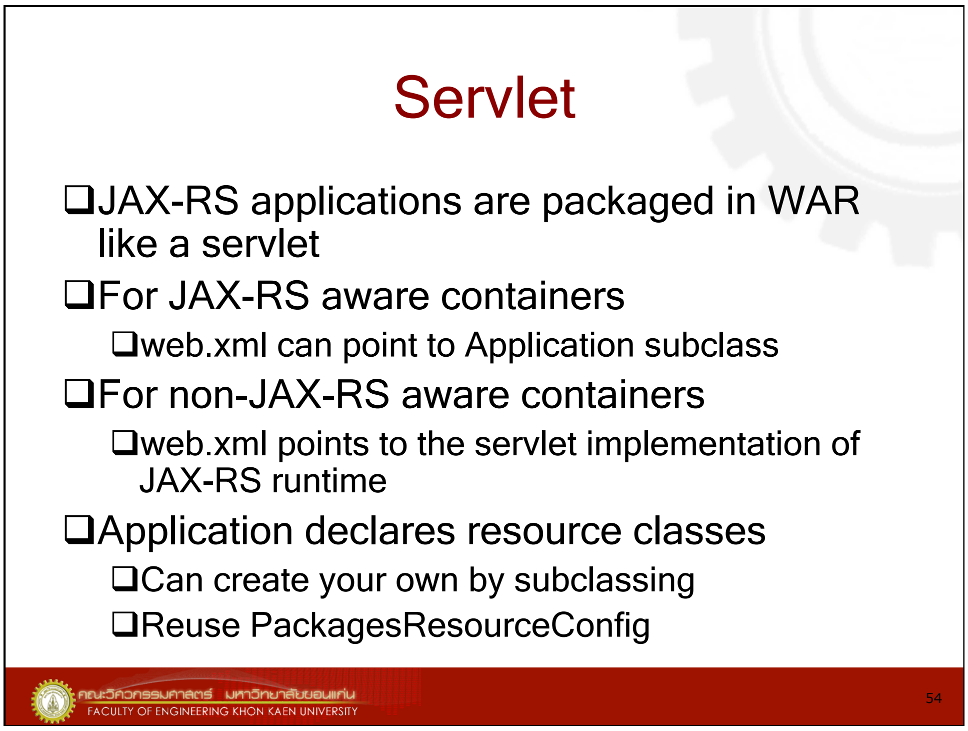
```
@Path("basket")
// Sub-resource locator could return a different resource if a
// user is a preferred customer:
public ShoppingBasketResource get(@Context SecurityContext
    sc) {
   if (sc.isUserInRole("PreferredCustomer") {
     return new PreferredCustomerShoppingBaskestResource();
   } else {
     return new ShoppingBasketResource();
   }
}
```



Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ **Deployment Options**
- ❑ Tools

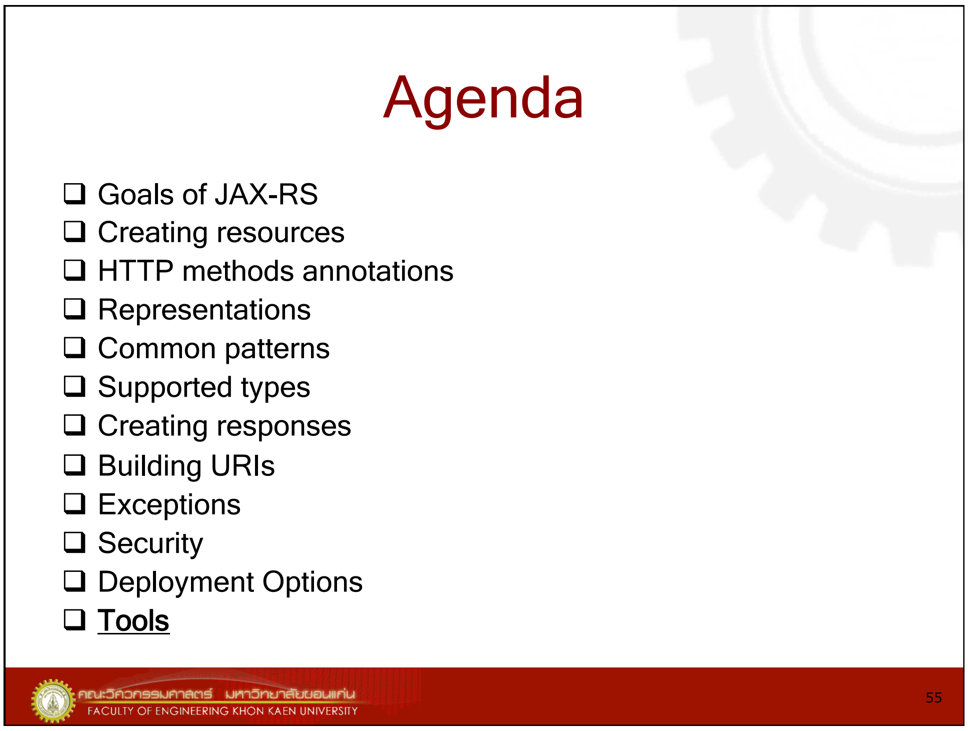
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 53



Servlet

- ❑ JAX-RS applications are packaged in WAR like a servlet
- ❑ For JAX-RS aware containers
  - ❑ web.xml can point to Application subclass
- ❑ For non-JAX-RS aware containers
  - ❑ web.xml points to the servlet implementation of JAX-RS runtime
- ❑ Application declares resource classes
  - ❑ Can create your own by subclassing
  - ❑ Reuse PackagesResourceConfig

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 54

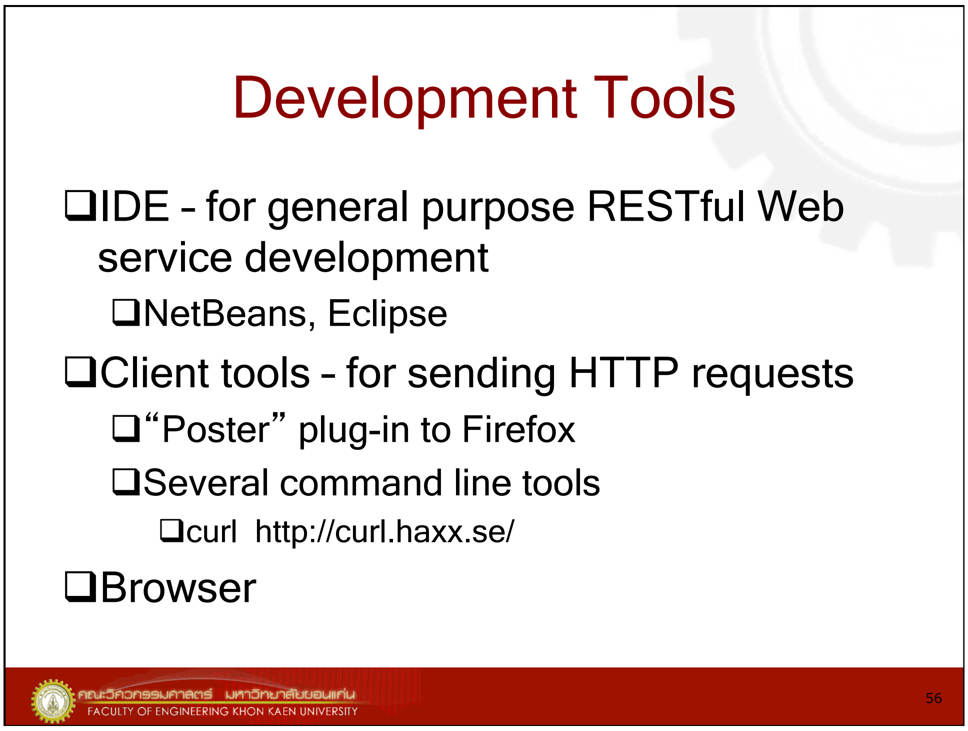


Agenda

- ❑ Goals of JAX-RS
- ❑ Creating resources
- ❑ HTTP methods annotations
- ❑ Representations
- ❑ Common patterns
- ❑ Supported types
- ❑ Creating responses
- ❑ Building URIs
- ❑ Exceptions
- ❑ Security
- ❑ Deployment Options
- ❑ Tools

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

55

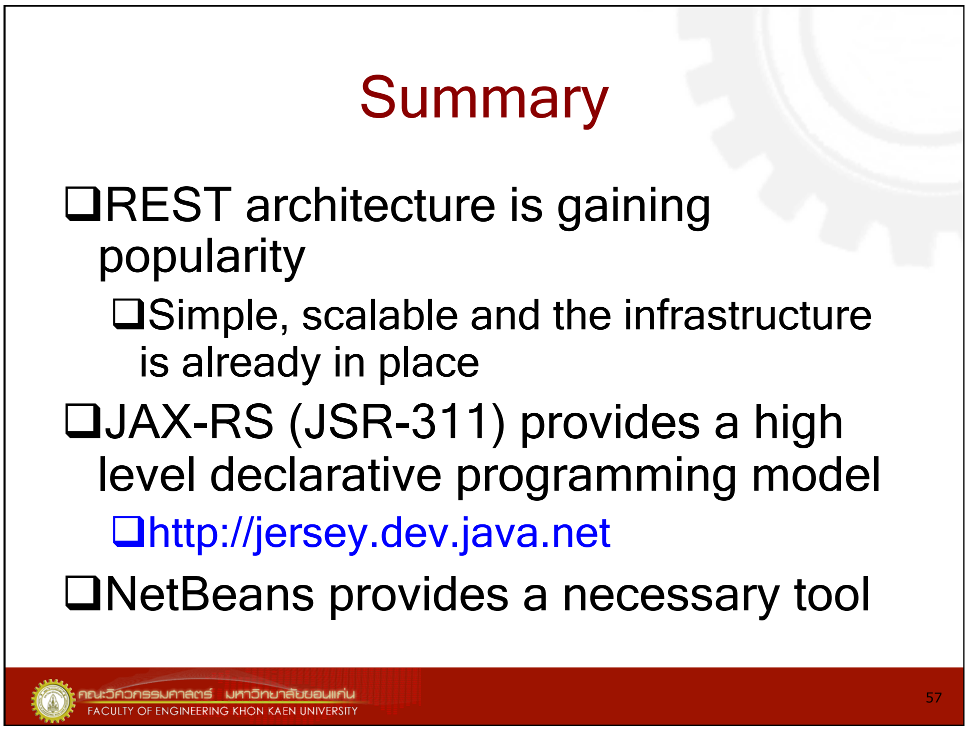


Development Tools

- ❑IDE - for general purpose RESTful Web service development
  - ❑NetBeans, Eclipse
- ❑Client tools - for sending HTTP requests
  - ❑“Poster” plug-in to Firefox
  - ❑Several command line tools
    - ❑curl http://curl.haxx.se/
- ❑Browser

FACULTY OF ENGINEERING KHON KAEN UNIVERSITY

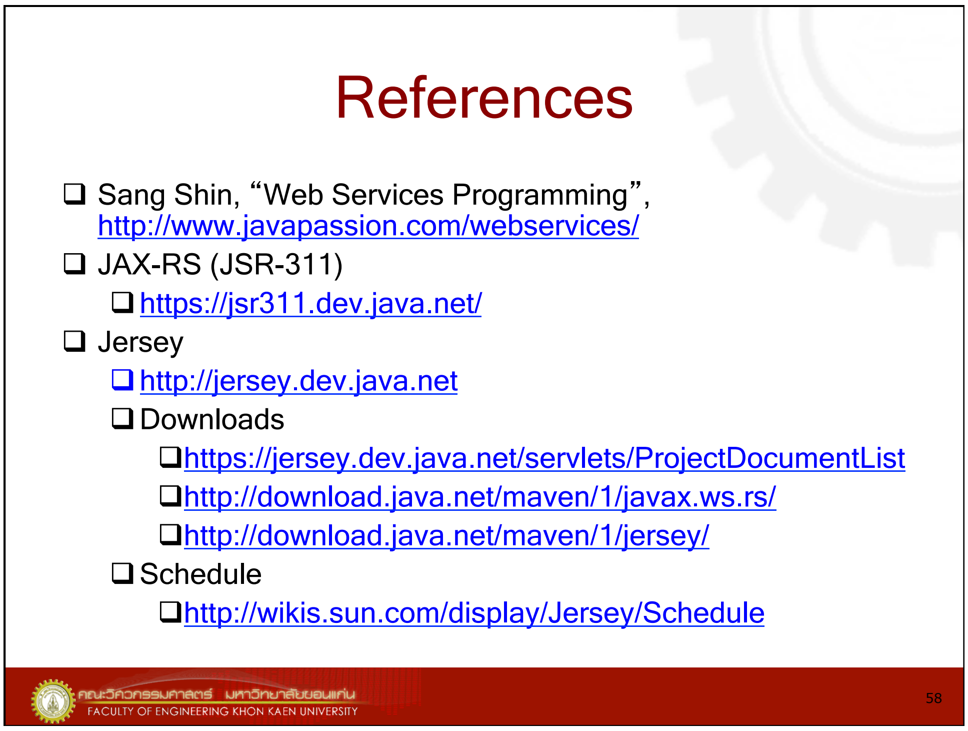
56



Summary

- REST architecture is gaining popularity
  - Simple, scalable and the infrastructure is already in place
- JAX-RS (JSR-311) provides a high level declarative programming model
  - http://jersey.dev.java.net
- NetBeans provides a necessary tool

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 57



References

- Sang Shin, "Web Services Programming", http://www.javapassion.com/webservices/
- JAX-RS (JSR-311)
  - https://jsr311.dev.java.net/
- Jersey
  - http://jersey.dev.java.net
  - Downloads
    - https://jersey.dev.java.net/servlets/ProjectDocumentList
    - http://download.java.net/maven/1/javax.ws.rs/
    - http://download.java.net/maven/1/jersey/
  - Schedule
    - http://wikis.sun.com/display/Jersey/Schedule

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
FACULTY OF ENGINEERING KHON KAEN UNIVERSITY 58