



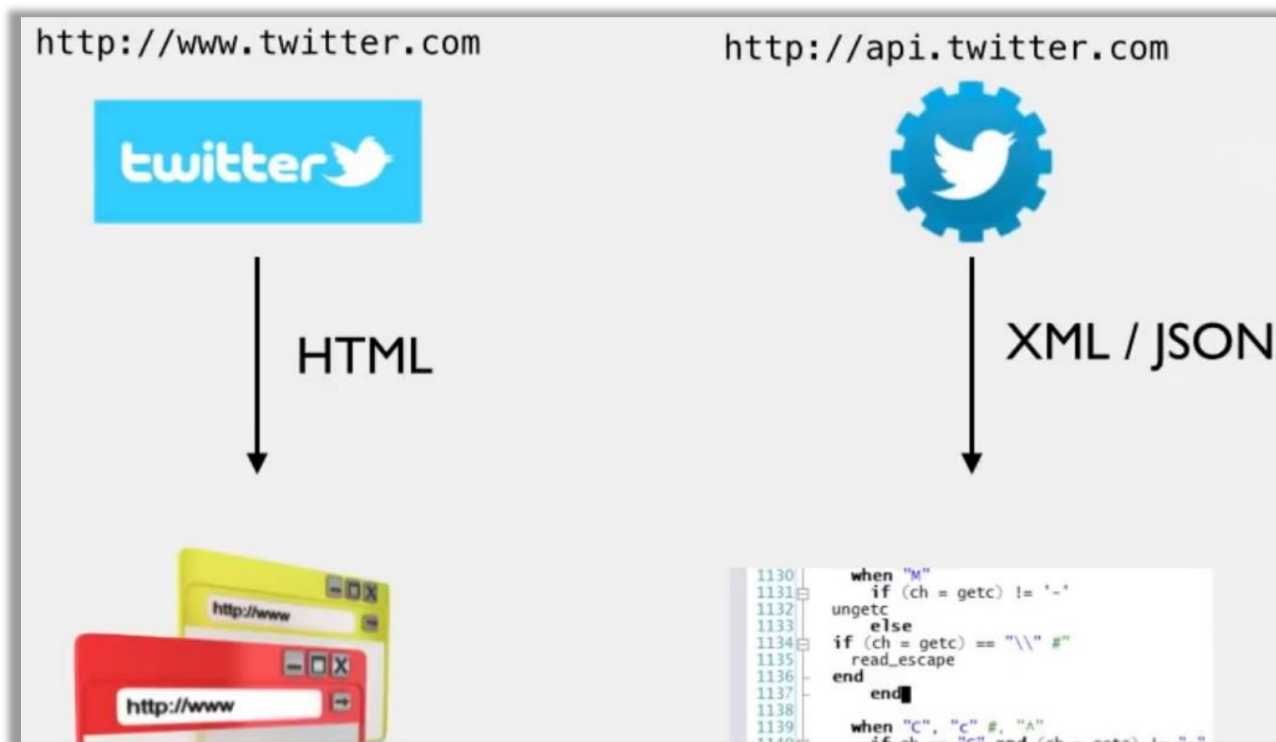
JAX-RS  Java EE

P-I

- هي مجموعة الخدمات التي تكون متاحة على الانترنت ليتم استخدامها برمجياً في تطوير التطبيقات.
- يمكن اعتبارها Online API يمكنك استدعائها من خلال برنامجك.

# What is Web Services

- يتم توفيرها عن طريق URL يتم تبادل المعلومات بصيف XML/JSON.



# Web Services Types:

---

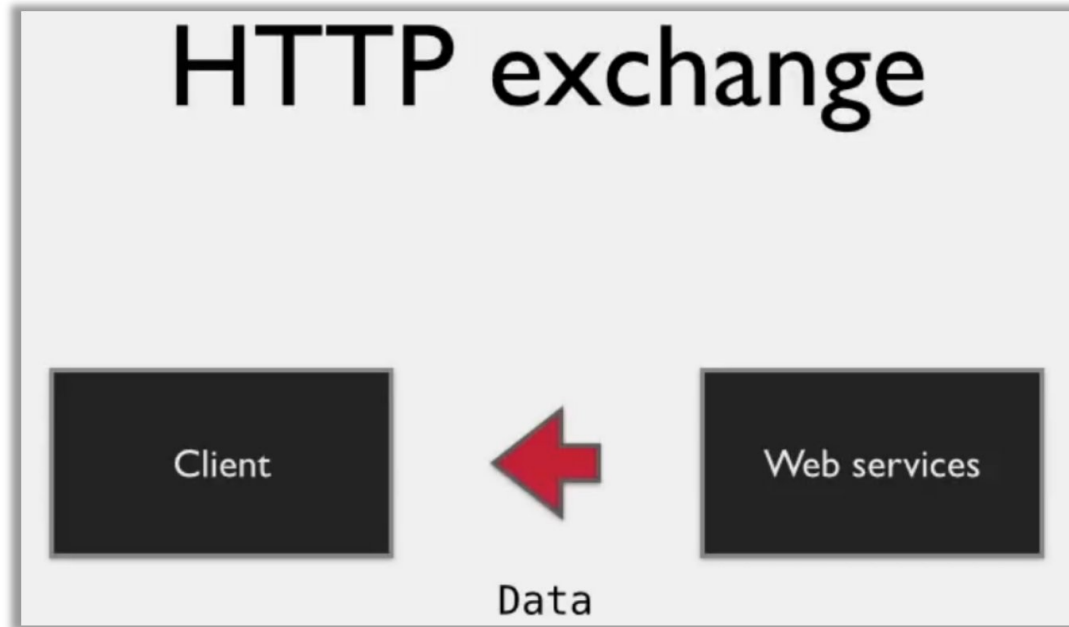


- RESTful Web Services

- SOAP Web Services

- 
- هي Web Service تتبع REST Architecture .
  - تسمح بتقديم API بصورة موحدة وأمنة.
  - مستخدم هذه API يمكن يجري مجموعة من العمليات باستخدام HTTP Methods .

- يتم تبادل المعلومات بين client و service بأستخدام HTTP protocol ويتم من خلاله تبادل البيانات بصيغ مختلفة مثل:



- XML

- JSON

- TEXT

# HTTP Methods

---



- GET •
- POST •
- PUT •
- DELETE •

- تستخدم لتبين نجاح HTTP Request من عدمه.

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone



- تستخدم لتبين نوع البيانات المرسله والمستقبل إلى السيرفر.

```
Content-Type: text/html; charset=UTF-8
```

```
Content-Type: multipart/form-data; boundary=something
```

## Resource Based URL

---

- تركز على resource الذي ستم العمل عليه.
- تتكون عادة من nouns.
- تعتمد على HTTP Methods لتحديد العمل الذي ستقدمه.

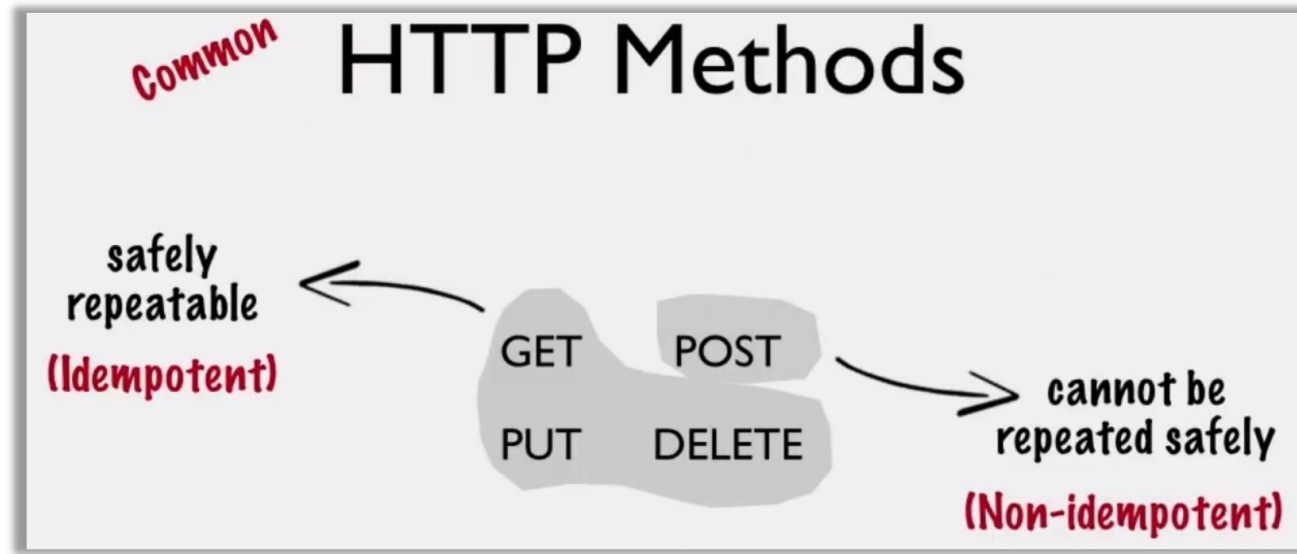
### Resource-based URL

---

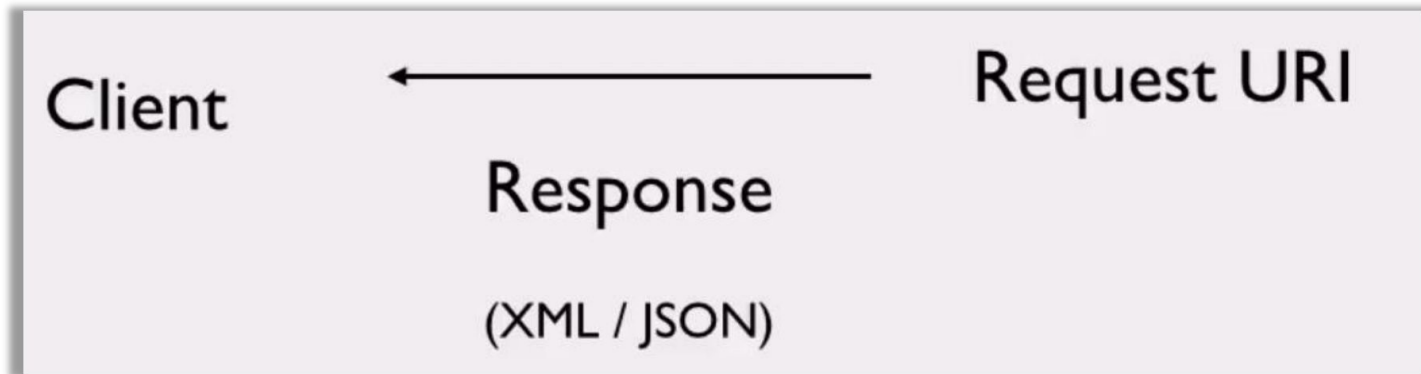
```
GET    /accounts/application
POST   /accounts
GET    /queries/form
POST   /queries
GET    /order/123
POST   /order/123/items
DELETE /order/123
```

# Method Idempotence

- هي إمكانية العملية عدة مرات دون ان تتغير النتيجة على ماكانت عليه اول مرة.



- عند استخدام REST الرد عادة يكون بيانات في أحد الصيغتين :
  - JSON
  - XML



# JSON (JavaScript Object Notation)

- هي تنسيق بسيط لنقل البيانات يسهل قراءته وكتابته من قبل المطورين كما يسهل تحليله وتوليده من قبل البرامج المستخدمة له.

```
{  
  "id": "10",  
  "message": "Hello world",  
  "created": "2014-06-01T18:06:36.902",  
  "author": "koushik"  
}
```

# XML (Extensible Markup Language)

- هي عبارة عن markup language تستخدم لتمثيل البيانات لتساعد في نقلها وتخزينها.

```
<messageEntity>  
  <id>10</id>  
  <message>Hello world</message>  
  <created>2014-06-01T18:06:36.902</created>  
  <author>koushik</author>  
</messageEntity>
```

- 
- هي عبارة JAVA EE Specification تقدم مجموعة من interfaces و annotation لتطوير Web Services من النوع REST .

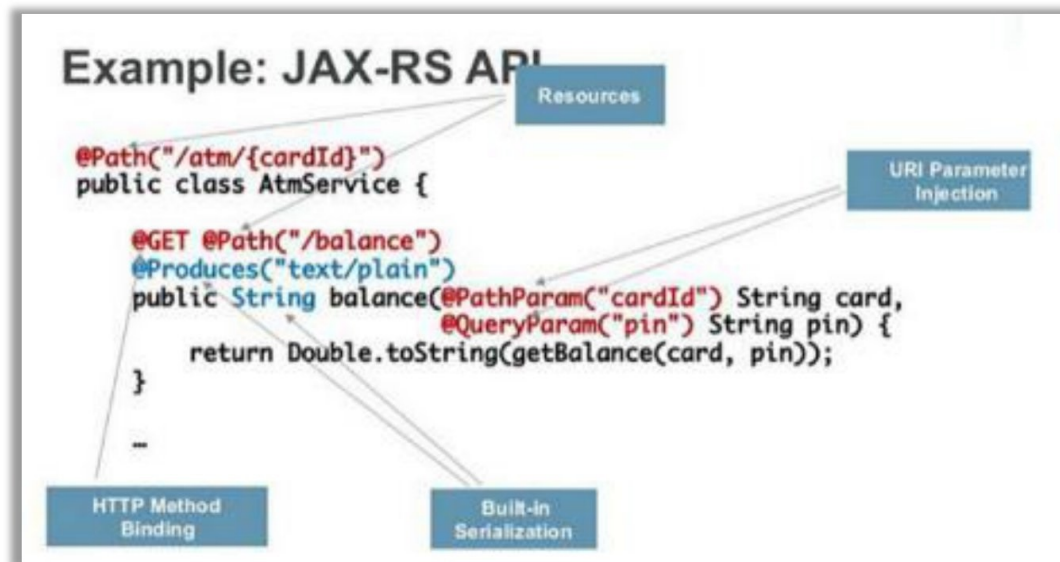
- تعتمد JAX-RS على annotation في تطوير Web Services والجدول التالي يبين أهمها:

JAX-RS Annotations			
HTTP Request Methods	Resource	Request-Response Media Types	Request Parameters
<ul style="list-style-type: none"><li>• @GET</li><li>• @POST</li><li>• @PUT</li><li>• @DELETE</li><li>• @HEAD</li><li>• @OPTIONS</li></ul>	<ul style="list-style-type: none"><li>• @Path</li></ul>	<ul style="list-style-type: none"><li>• @Produces</li><li>• @Consumes</li></ul>	<ul style="list-style-type: none"><li>• @PathParam</li><li>• @QueryParam</li><li>• @MatrixParam</li><li>• @HeaderParam</li><li>• @CookieParam</li><li>• @DefaultValue</li><li>• @Context</li><li>• @BeanParam</li><li>• @Encoded</li></ul>



# Example

- الشكل التالي يبين استخدام JAX-RS annotations لتطوير Web Service .



## REST Resource

---

- هو عبارة عن Object له نوع وبيانات مرتبطة وعلاقات مع resources الأخرى ومجموعة من method التي تعمل عليه. ويمكن تجميعه في مجموعات متجانسة تعرف ب Collection .

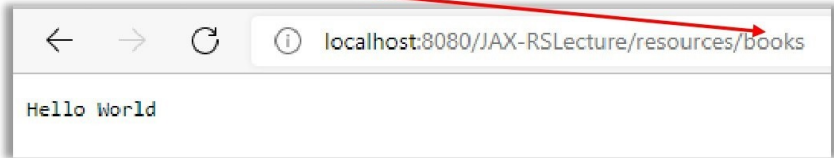
# @Path

- عند انشاء resource تستخدم @Path annotation لعمل mapping له مع HTTP Request .URI

```
@Path("books")
public class BooksResource {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getBooks() {

        return "Hello World";
    }
}
```



- تستخدم لتحديد أن هذه الخدمة سيتم طلبها عن طريق HTTP GET method .

```
@Path("books")
public class BooksResource {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getBooks() {

        return "Hello World";
    }
}
```

## @Produces

---

- تستخدم لتحديد نوع response التي ترجعه Web Service .
- للحصول على رد من النوع XML نستخدم `@Produces(MediaType.APPLICATION_XML)`
- للحصول على رد من النوع JSON نستخدم `@Produces(MediaType.APPLICATION_JSON)`
- للحصول على رد من النوع TEXT نستخدم `@Produces(MediaType.TEXT_PLAIN)`

# XML Response



```
@Path("books")
public class BooksResource {

    private List<Book> books;

    @GET
    @Produces(MediaType.APPLICATION_XML)
    public List<Book> getBooks() {
        books = new ArrayList<>();
        books.add(new Book("Learning Java ", "Salem", 35.5f, new Date(), 320, "History"));
        books.add(new Book("Learning C ", "Ali", 35.5f, new Date(), 450, "Comics"));
        books.add(new Book("Learning Python ", "Khalid", 35.5f, new Date(), 230, "Scifi"));
        return books;
    }
}
```

localhost:8080/JAX-RSLecture/resources/books

This XML file does not appear to have any style information associated with it.

```
<books>
  <book>
    <author>Salem</author>
    <number>320</number>
    <price>35.5</price>
    <publishingDate>2022-04-10T07:25:54.435+02:00</publishingDate>
    <title>Learning Java </title>
    <type>History</type>
  </book>
  <book>
    <author>Ali</author>
    <number>450</number>
    <price>35.5</price>
    <publishingDate>2022-04-10T07:25:54.435+02:00</publishingDate>
    <title>Learning C </title>
    <type>Comics</type>
  </book>
  <book>
    <author>Khalid</author>
    <number>230</number>
    <price>35.5</price>
    <publishingDate>2022-04-10T07:25:54.435+02:00</publishingDate>
    <title>Learning Python </title>
    <type>Scifi</type>
  </book>
</books>
```

# @XmlElement

- يتم من خلالها عمل mapping بين Class و XML Root مما يمكننا من الرد بالصيغ XML.

```
import javax.xml.bind.annotation.XmlRootElement;

/**
 * @author elwae
 */
@XmlRootElement
public class Book {

    private String title;
    private String author;
    private Float price;
    private Date publishingDate;
    private int number;
    private String type;

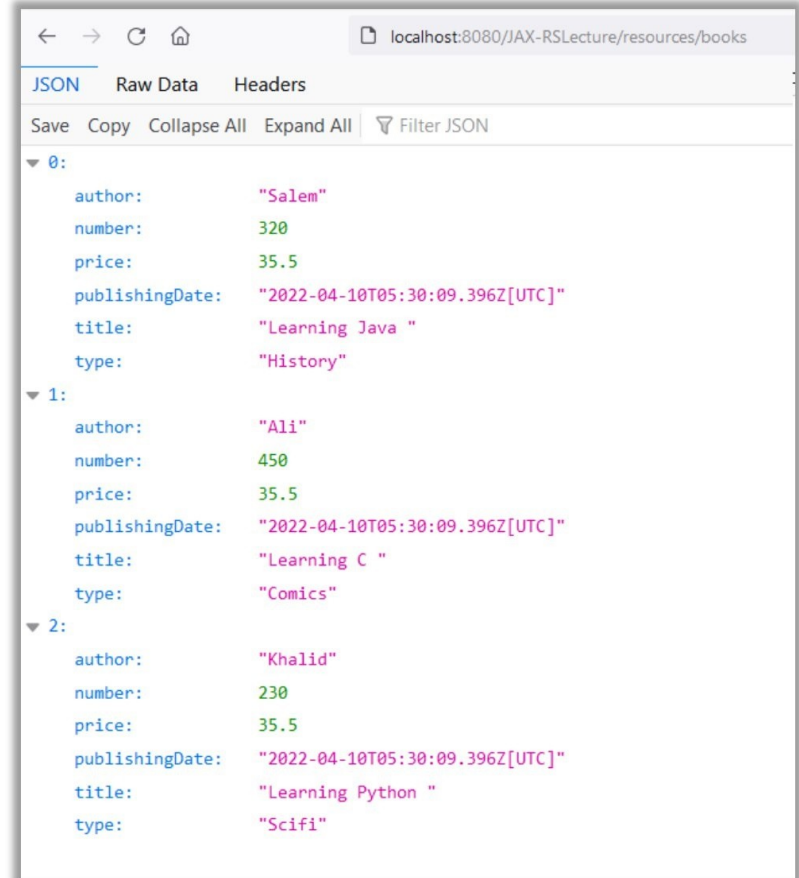
    public Book() {
    }
}
```

# JSON Response

```
@Path("books")
public class BooksResource {

    private List<Book> books;

    @GET
    @Produces(MediaType.APPLICATION_JSON)
    public List<Book> getBooks() {
        books = new ArrayList<>();
        books.add(new Book("Learning Java ", "Salem", 35.5f, new Date(), 320, "History"));
        books.add(new Book("Learning C ", "Ali", 35.5f, new Date(), 450, "Comics"));
        books.add(new Book("Learning Python ", "Khalid", 35.5f, new Date(), 230, "Scifi"));
        return books;
    }
}
```

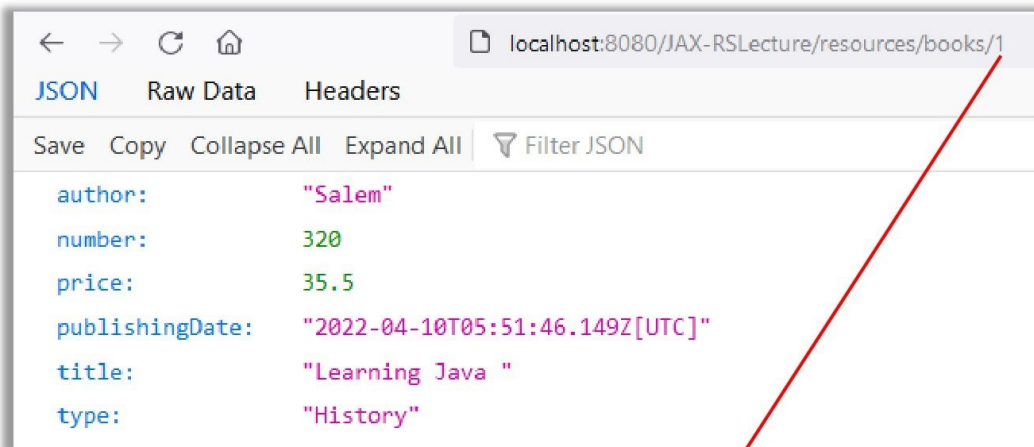


The screenshot shows a web browser window with the URL `localhost:8080/JAX-RSLecture/resources/books`. The browser is displaying the JSON response for the `books` resource. The response is a JSON array with three elements, each representing a book. The browser interface includes navigation buttons, a tab for the current page, and a menu with options like 'JSON', 'Raw Data', and 'Headers'. Below the menu, there are buttons for 'Save', 'Copy', 'Collapse All', 'Expand All', and a 'Filter JSON' dropdown. The JSON data is displayed in a tree view, with each book object expanded to show its fields: `author`, `number`, `price`, `publishingDate`, `title`, and `type`.

```
{
  "0": {
    "author": "Salem",
    "number": 320,
    "price": 35.5,
    "publishingDate": "2022-04-10T05:30:09.396Z[UTC]",
    "title": "Learning Java ",
    "type": "History"
  },
  "1": {
    "author": "Ali",
    "number": 450,
    "price": 35.5,
    "publishingDate": "2022-04-10T05:30:09.396Z[UTC]",
    "title": "Learning C ",
    "type": "Comics"
  },
  "2": {
    "author": "Khalid",
    "number": 230,
    "price": 35.5,
    "publishingDate": "2022-04-10T05:30:09.396Z[UTC]",
    "title": "Learning Python ",
    "type": "Scifi"
  }
}
```



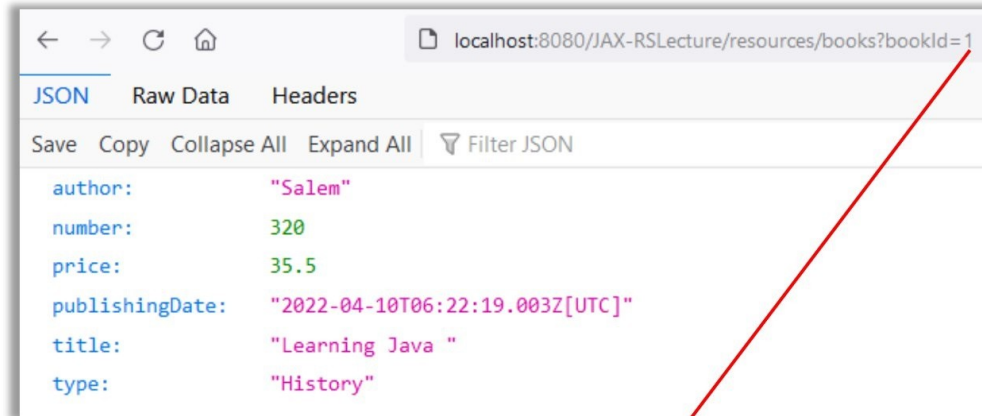
- تسمح بالحصول على القيمة الموجودة في URI template .



```
localhost:8080/JAX-RSLecture/resources/books/1
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
author: "Salem"
number: 320
price: 35.5
publishingDate: "2022-04-10T05:51:46.149Z[UTC]"
title: "Learning Java "
type: "History"
```

```
@GET
@Path("/{bookId}")
@Produces(MediaType.APPLICATION_JSON)
public Book getBook(@PathParam("bookId") int bookId) {
    bookMap = new HashMap<>();
    bookMap.put(1, new Book("Learning Java ", "Salem", 35.5f, new Date(), 320, "History"));
    bookMap.put(2, new Book("Learning C ", "Ali", 35.5f, new Date(), 450, "Comics"));
    bookMap.put(3, new Book("Learning Python ", "Khalid", 35.5f, new Date(), 230, "Scifi"));
    return bookMap.get(bookId);
}
```

- تستخدم للحصول على القيمة الموجودة في URI Query Parameter .



```
@GET
@Produces(MediaType.APPLICATION_JSON)
public Book getBook(@QueryParam("bookId") int bookId) {
    bookMap = new HashMap<>();
    bookMap.put(1, new Book("Learning Java ", "Salem", 35.5f, new Date(), 320, "History"));
    bookMap.put(2, new Book("Learning C ", "Ali", 35.5f, new Date(), 450, "Comics"));
    bookMap.put(3, new Book("Learning Python ", "Khalid", 35.5f, new Date(), 230, "Scifi"));
    return bookMap.get(bookId);
}
```