

SoftUni Team
Technical Trainers

Software University

<http://softuni.bg>

JSON

Exporting and Importing Data from JSON format

*Databases
Frameworks*

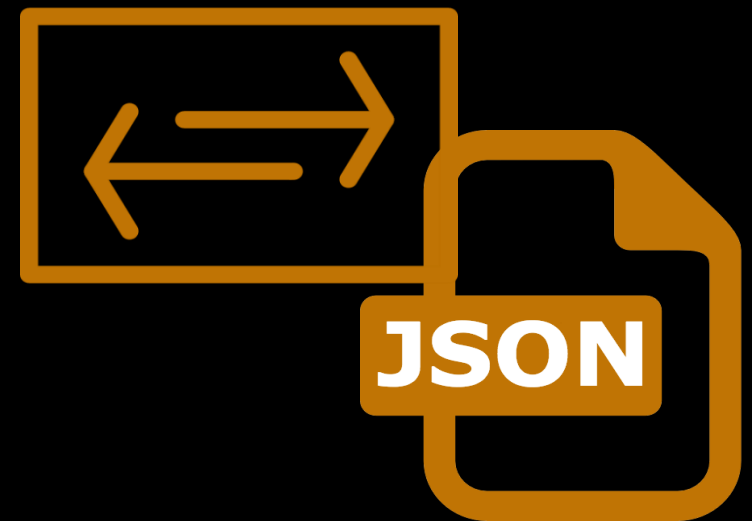




Table of Contents




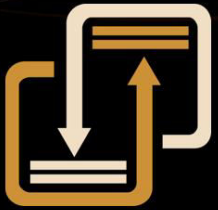


JSON

Transmitting data objects via attribute-value pairs

4





GSON

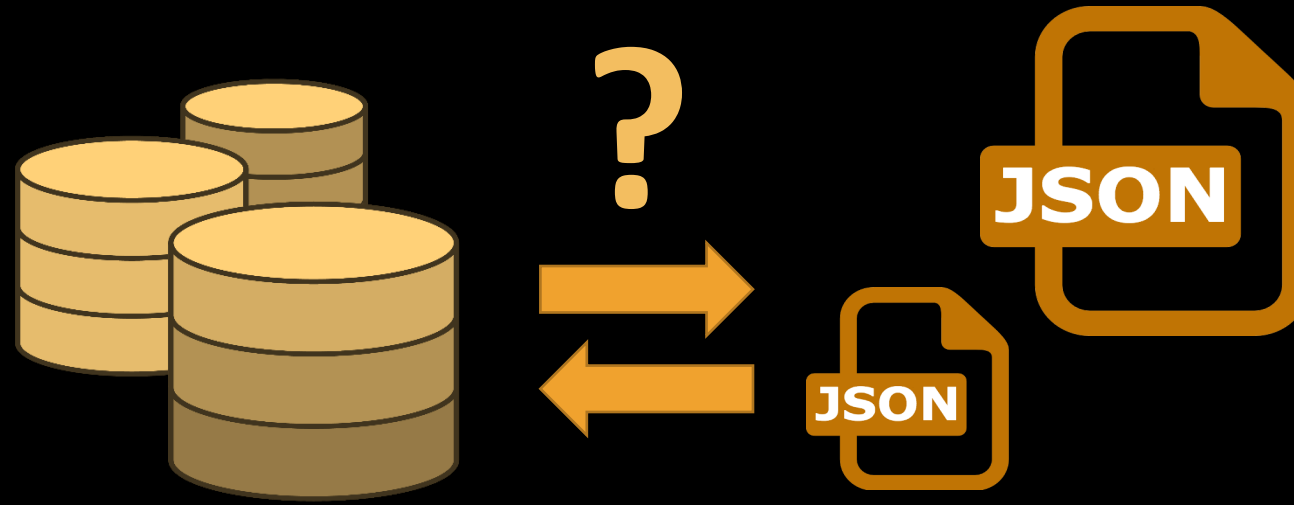
Serialize and deserialize objects with Java

10



sli.do

#db-advanced



JSON

Transmitting data objects via attribute-
value pairs

- JavaScript Object Notation
 - Human-readable format to transmit **data objects** consisting of **attribute–value pairs** and **arrays**
 - Subset of JavaScript syntax
- Supports several data types:
 - Number, String, Boolean, Array, Object, null

JSON Example

person.json

```
{  
  "firstName": "Daniel",  
  "lastName": "Sempere",  
  "age": 24,  
  "isMarried": true  
}
```

Key

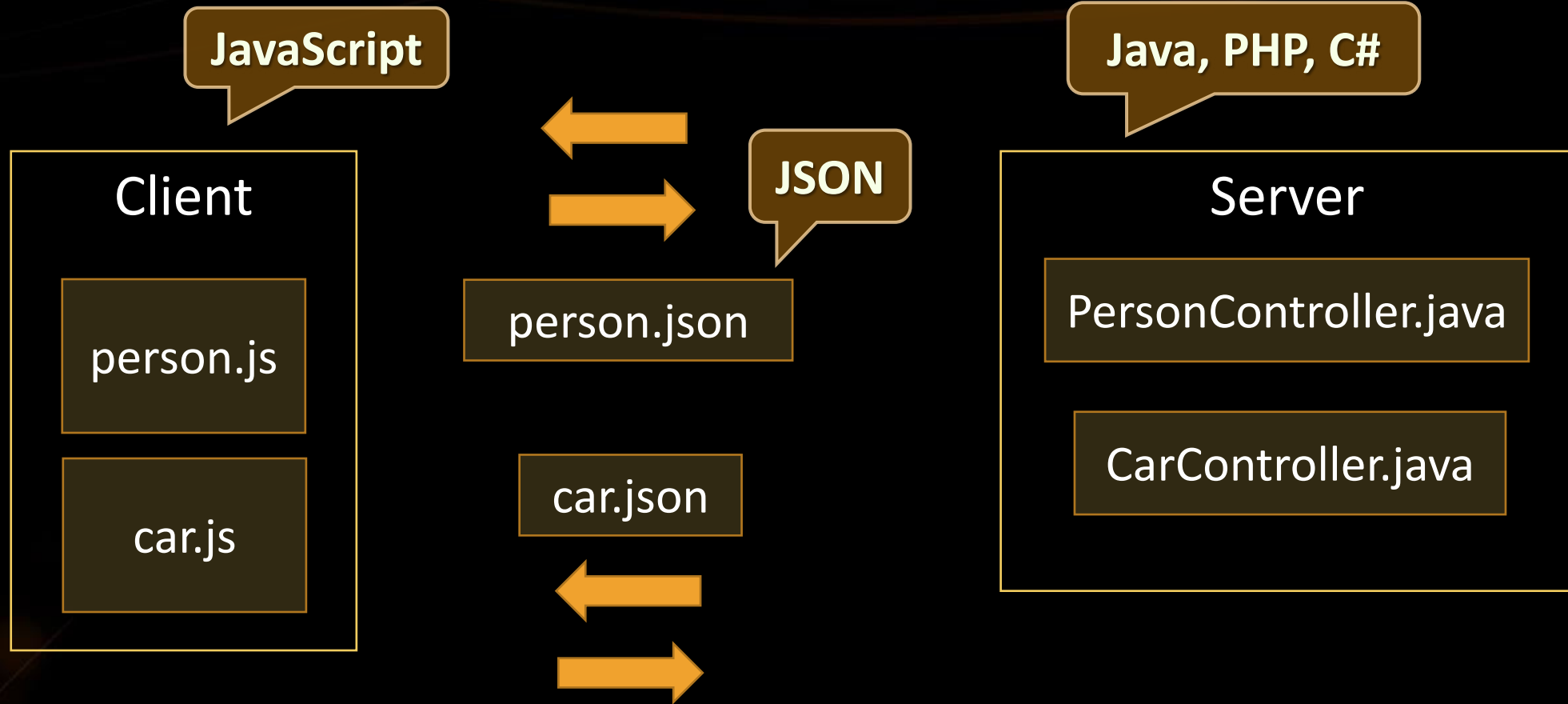
Value

student.json

```
{  
  "firstName": "Daniel",  
  "lastName": "Sempere",  
  "age": 24,  
  "courses": [  
    {  
      "name": "Java DB",  
    },  
    {  
      "name": "HTML",  
    },  
  ],  
}
```

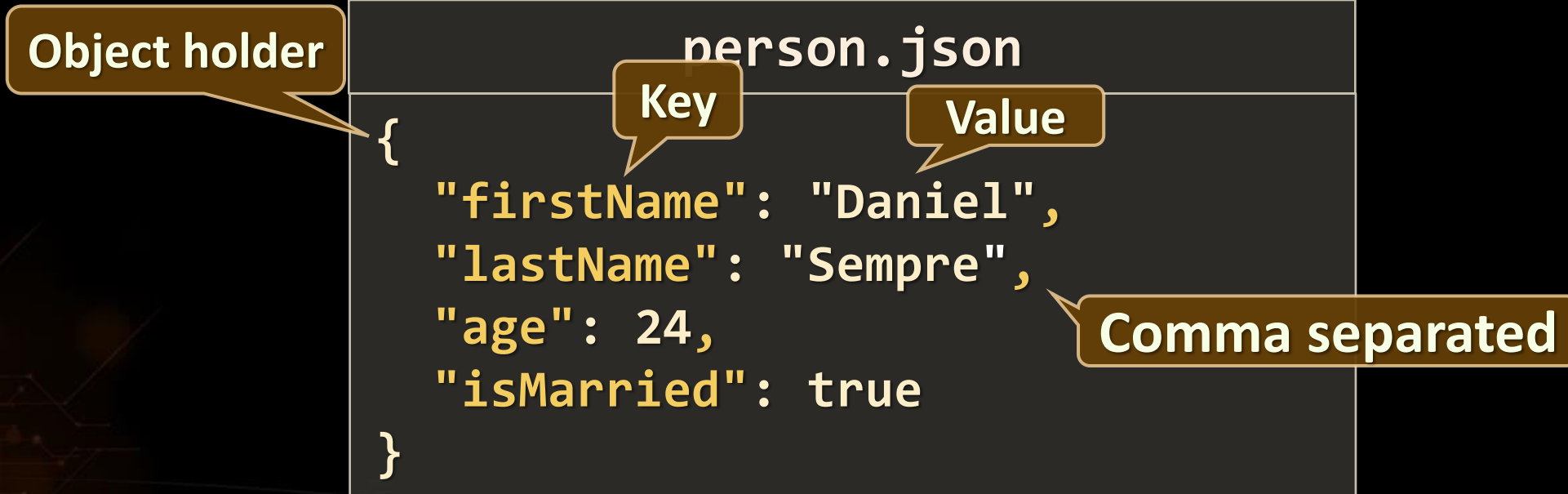
Array type value

JSON Function



JSON Structure

- Data is represented in **name/value** pairs
- Curly braces hold objects
- Square brackets hold **arrays**



JSON Structure





GSON

Serialize and deserialize objects with
Java

- Provides easy to use mechanisms to convert Java to JSON and vice-versa
 - Originally developed by Google
- Generate compact and readability JSON output

`pom.xml`

```
<dependency>  
    <groupId>com.google.code.gson</groupId>  
    <artifactId>gson</artifactId>  
</dependency>
```

GSON Initialization

- Gson objects are responsible for the JSON manipulations
 - GsonBuilder creates an instance of GSON
 - `excludeFieldsWithoutExposeAnnotation()` – excludes fields without `@Expose` annotation
 - `setPrettyPrinting()` – aligns and justifies the created JSON format
 - `create()` – creates an instance of Gson

JsonParser.java

```
Gson gson = new GsonBuilder()  
    .excludeFieldsWithoutExposeAnnotation()  
    .setPrettyPrinting()  
    .create();
```

Export Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto implements Serializable {  
  
    @Expose  
    private String country;  
  
    @Expose  
    private String city;  
  
    @Expose  
    private String street;  
  
}
```

The field will be
imported/exported

JsonParser.java

```
AddressJsonDto addressJsonDto = new AddressJsonDto();  
addressJsonDto.setCountry("Bulgaria");  
addressJsonDto.setCity("Sofia");  
addressJsonDto.setStreet("Mladost 4");  
String content = this.gson.toJson(addressJsonDto);
```

Creates JSON

Export Single Object to JSON

JsonParser.java

```
AddressJsonDto addressJsonDto = new AddressJsonDto();  
addressJsonDto.setCountry("Bulgaria");  
addressJsonDto.setCity("Sofia");  
addressJsonDto.setStreet("Mladost 4");  
String content = this.gson.toJson(addressJsonDto);
```

address.json

```
{  
  "country": "Bulgaria",  
  "city": "Sofia",  
  "street": "Mladost 4"  
}
```

Export Multiple Object to JSON

JsonParser.java

```
List<AddressJsonDto> addressJsonDtos = new ArrayList<>();  
addressJsonDtos.add(addressJsonDtoBulgaria);  
addressJsonDtos.add(addressJsonDtoSpain);  
String content = this.gson.toJson(addressJsonDtos);
```

addresses.json

```
[  
  {  
    "country": "Bulgaria",  
    "city": "Sofia",  
    "street": "Mladost 4"  
  },  
  {  
    "country": "Spain",  
    "city": "Barcelona",  
    "street": "Las Ramblas"  
  }  
]
```

Import Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto implements Serializable {  
  
    @Expose  
    private String country;  
  
    @Expose  
    private String city;  
  
    @Expose  
    private String street;  
  
}
```

The field will be
imported/exported

JsonParser.java

```
AddressJsonDto addressJsonDto =  
    this.gson.fromJson(AddressJsonDto.class, "/files/input/json/address.json");
```

Import Single Object to JSON

AddressJsonDto.java

```
public class AddressJsonDto  
implements Serializable {
```

```
    @Expose
```

```
    private String country;
```

```
    @Expose
```

```
    private String city;
```

```
    @Expose
```

```
    private String street;
```

```
}
```

address.json

```
{  
  "country": "Bulgaria",  
  "city": "Sofia",  
  "street": "Mladost 4"  
}
```

Import Multiple Object to JSON

JsonParser.java

```
AddressJsonDto[] addressJsonDtos =  
    this.gson.fromJson(AddressJsonDto[].class, "/files/input/json/addresses.json");
```

Object Array

addresses.json

```
[  
  {  
    "country": "Bulgaria",  
    "city": "Sofia",  
    "street": "Mladost 4"  
  },  
  {  
    "country": "Spain",  
    "city": "Barcelona",  
    "street": "Las Ramblas"  
  }  
]
```


Summary

- JSON is a very easy to use and understand format
- GSON is a java library to operate with JSON files
 - Easy import and export



JSON



Questions?

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license



- Attribution: this work may contain portions from
 - "Databases" course by Telerik Academy under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University Forums
 - forum.softuni.bg

