

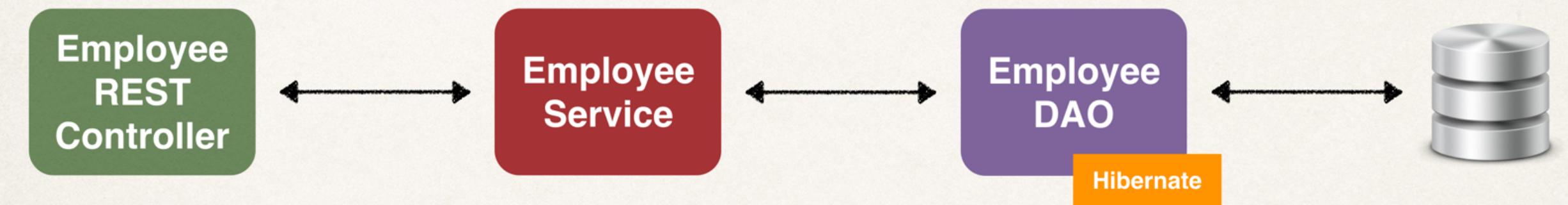
Rest Controller Methods - Find and Add



Development Process

Step-By-Step

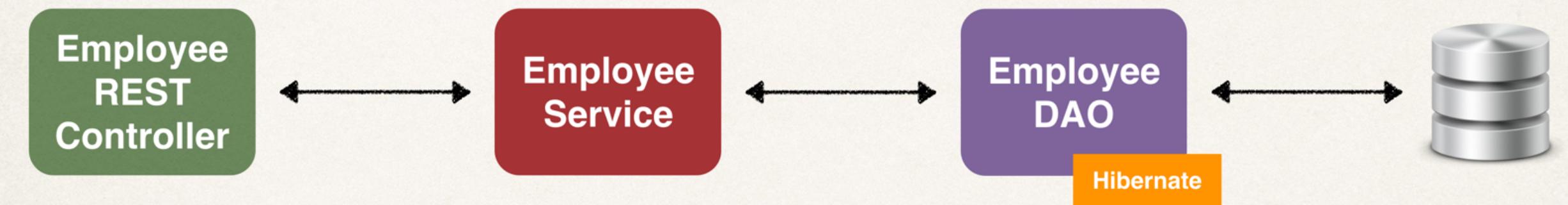
1. Set up Database Dev Environment
2. Create Spring Boot project using Spring Initializr
3. Get list of employees
4. Get single employee by ID
5. Add a new employee
6. Update an existing employee
7. Delete an existing employee



Development Process

Step-By-Step

1. Set up Database Dev Environment
2. Create Spring Boot project using Spring Initializr
3. Get list of employees
4. Get single employee by ID
5. Add a new employee
6. Update an existing employee
7. Delete an existing employee



Development Process

Step-By-Step

1. Set up Database Dev Environment
2. Create Spring Boot project using Spring Initializr
3. Get list of employees
4. Get single employee by ID
5. Add a new employee
6. Update an existing employee
7. Delete an existing employee

Rest Controller
methods



Real-Time Project

HTTP Method		CRUD Action
GET	/api/employees	<u>Read a list of employees</u>
GET	/api/employees/{employeeId}	<u>Read a single employee</u>
POST	/api/employees	<u>Create a new employee</u>
PUT	/api/employees	<u>Update an existing employee</u>
DELETE	/api/employees/{employeeId}	<u>Delete an existing employee</u>

Real-Time Project

Checkpoint

HTTP Method		CRUD Action
GET	/api/employees	<u>Read a list of employees</u>
GET	/api/employees/{employeeId}	<u>Read a single employee</u>
POST	/api/employees	<u>Create a new employee</u>
PUT	/api/employees	<u>Update an existing employee</u>
DELETE	/api/employees/{employeeId}	<u>Delete an existing employee</u>

Real-Time Project

Checkpoint

HTTP Method		CRUD Action
 GET	/api/employees	<u>Read a list of employees</u>
GET	/api/employees/{employeeId}	<u>Read a single employee</u>
POST	/api/employees	<u>Create a new employee</u>
PUT	/api/employees	<u>Update an existing employee</u>
DELETE	/api/employees/{employeeId}	<u>Delete an existing employee</u>

Real-Time Project

Checkpoint

HTTP Method		CRUD Action
 GET	/api/employees	<u>Read a list of employees</u>
 GET	/api/employees/{employeeId}	<u>Read a single employee</u>
POST	/api/employees	<u>Create a new employee</u>
PUT	/api/employees	<u>Update an existing employee</u>
DELETE	/api/employees/{employeeId}	<u>Delete an existing employee</u>

Real-Time Project

Checkpoint

HTTP Method		CRUD Action
 GET	/api/employees	<u>Read a list of employees</u>
 GET	/api/employees/{employeeId}	<u>Read a single employee</u>
 POST	/api/employees	<u>Create a new employee</u>
PUT	/api/employees	<u>Update an existing employee</u>
DELETE	/api/employees/{employeeId}	<u>Delete an existing employee</u>

Read a Single Employee

Read a Single Employee

REST
Client

Read a Single Employee

REST
Client

Employee
REST
Controller

Read a Single Employee

REST
Client

GET

/api/employees/{employeeId}



Employee
REST
Controller

Read a Single Employee



Employee
REST
Controller

Create a New Employee

REST
Client

Employee
REST
Controller

Create a New Employee

REST
Client

POST

/api/employees

```
{  
    "firstName": "Juan",  
    "lastName": "Perez",  
    "email": "juan.perez@luv2code.com"  
}
```

Employee
REST
Controller

Create a New Employee

Since new employee,
we are not
passing id / primary key

REST
Client

POST /api/employees

```
{  
    "firstName": "Juan",  
    "lastName": "Perez",  
    "email": "juan.perez@luv2code.com"  
}
```

Employee
REST
Controller

Create a New Employee

Since new employee,
we are not
passing id / primary key

REST
Client

POST /api/employees

```
{  
  "firstName": "Juan",  
  "lastName": "Perez",  
  "email": "juan.perez@luv2code.com"  
}
```

Employee
REST
Controller



```
{  
  "id": 7,  
  "firstName": "Juan",  
  "lastName": "Perez",  
  "email": "juan.perez@luv2code.com"  
}
```

Create a New Employee

Since new employee,
we are not
passing id / primary key

REST
Client

POST

/api/employees

```
{  
    "firstName": "Juan",  
    "lastName": "Perez",  
    "email": "juan.perez@luv2code.com"  
}
```

Employee
REST
Controller

Response contains
new id / primary key

```
{  
    "id": 7,  
    "firstName": "Juan",  
    "lastName": "Perez",  
    "email": "juan.perez@luv2code.com"  
}
```

Sending JSON to Spring REST Controllers

Sending JSON to Spring REST Controllers

- When sending JSON data to Spring REST Controllers

Sending JSON to Spring REST Controllers

- When sending JSON data to Spring REST Controllers
- For controller to process JSON data, need to set a HTTP request header

Sending JSON to Spring REST Controllers

- When sending JSON data to Spring REST Controllers
- For controller to process JSON data, need to set a HTTP request header
 - Content-type: application/json

Sending JSON to Spring REST Controllers

- When sending JSON data to Spring REST Controllers
- For controller to process JSON data, need to set a HTTP request header
 - Content-type: application/json
- Need to configure REST client to send the correct HTTP request header

Postman - Sending JSON in Request Body

Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman

Postman - Sending JSON in Request Body

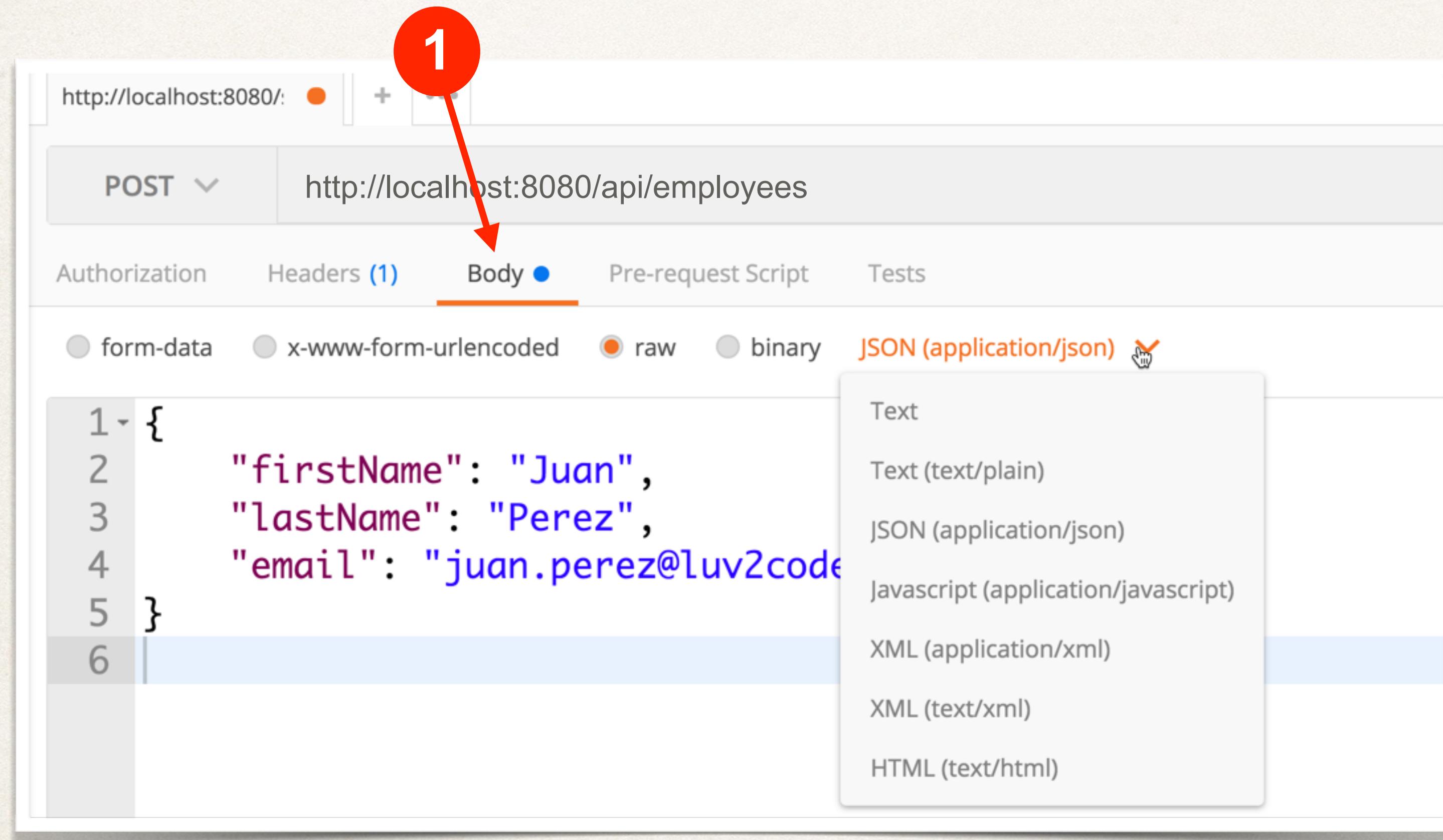
- Must set HTTP request header in Postman

The screenshot shows the Postman application interface. At the top, there is a header bar with a search field containing "http://localhost:8080/" and several icons. Below this is a main area with a "POST" method selected and a URL "http://localhost:8080/api/employees". Underneath, there are tabs for "Authorization", "Headers (1)", "Body" (which is active and highlighted in blue), and "Pre-request Script" and "Tests". The "Body" tab has four options: "form-data", "x-www-form-urlencoded", "raw" (which is selected and highlighted in orange), and "binary". A dropdown menu is open over the "raw" option, listing "Text", "Text (text/plain)", "JSON (application/json)" (which is also highlighted in orange), "Javascript (application/javascript)", "XML (application/xml)", "XML (text/xml)", and "HTML (text/html)". In the main body area, there is a code editor with the following JSON payload:

```
1 {  
2   "firstName": "Juan",  
3   "lastName": "Perez",  
4   "email": "juan.perez@luv2code.com"  
5 }  
6
```

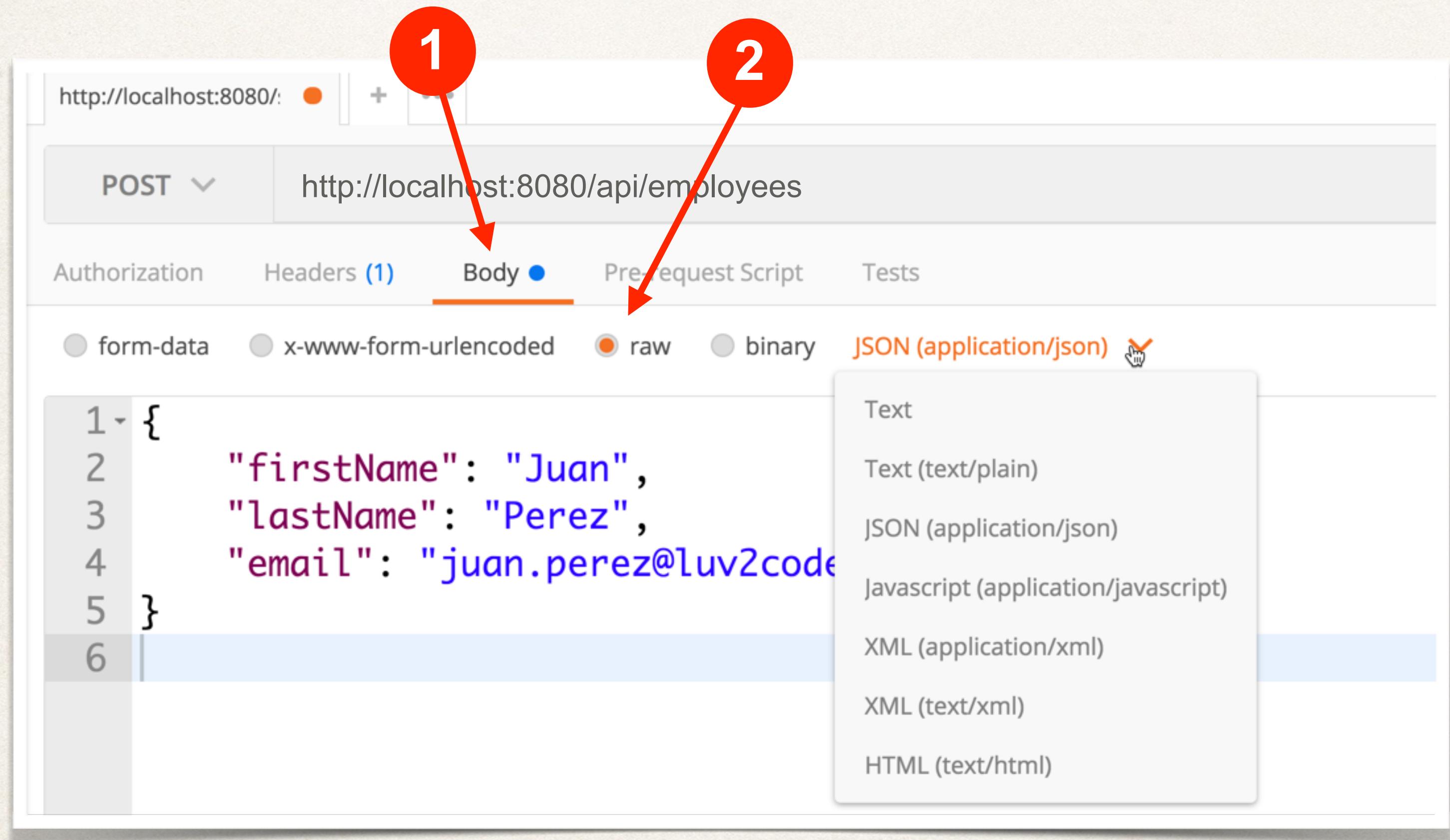
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



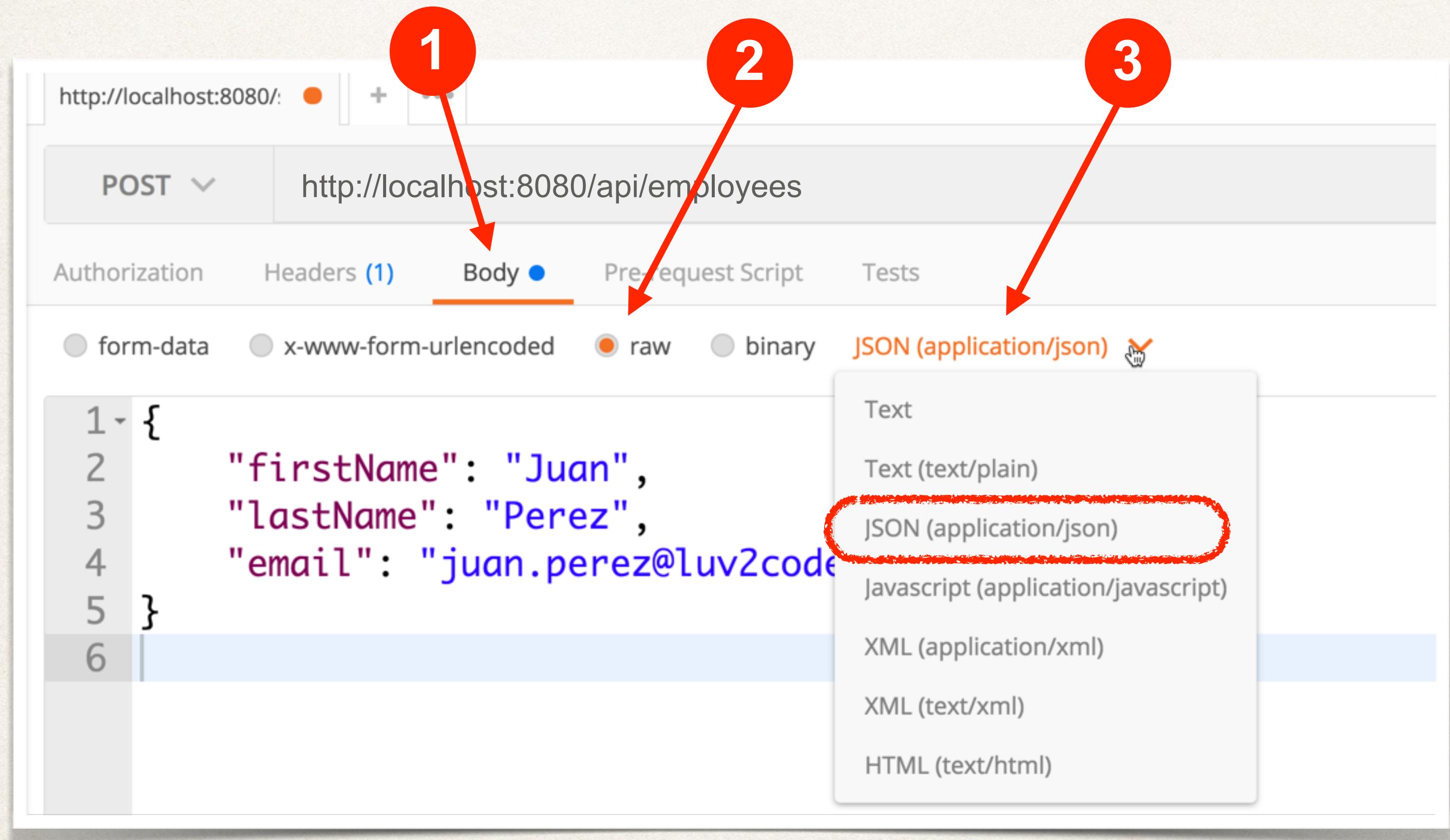
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/api/employees`. The 'Body' tab is selected (indicated by a red arrow labeled 1). The dropdown menu below shows various options: Text, Text (text/plain), **JSON (application/json)** (circled in red), Javascript (application/javascript), XML (application/xml), XML (text/xml), and HTML (text/html). The JSON payload in the body is:

```
1 {  
2   "firstName": "Juan",  
3   "lastName": "Perez",  
4   "email": "juan.perez@luv2code.com"  
5 }  
6
```

**Based on these configs,
Postman will automatically set
the correct HTTP request header**