

Spring CRM REST - Add Customer



Real-Time Project

HTTP Method		CRUD Action
POST	/api/customers	<u>C</u> reate a new customer
GET	/api/customers	<u>R</u> ead a list of customers
GET	/api/customers/{customerId}	<u>R</u> ead a single customer
PUT	/api/customers	<u>U</u> pdate an existing customer
DELETE	/api/customers/{customerId}	<u>D</u> delete an existing customer

Real-Time Project

Checkpoint

HTTP Method		CRUD Action
POST	/api/customers	<u>C</u> reate a new customer
GET	/api/customers	<u>R</u> ead a list of customers
GET	/api/customers/{customerId}	<u>R</u> ead a single customer
PUT	/api/customers	<u>U</u> pdate an existing customer
DELETE	/api/customers/{customerId}	<u>D</u> delete an existing customer




Real-Time Project

Checkpoint

HTTP Method		CRUD Action
POST	/api/customers	<u>C</u> reate a new customer
✓ GET	/api/customers	<u>R</u> ead a list of customers
✓ GET	/api/customers/{customerId}	<u>R</u> ead a single customer
PUT	/api/customers	<u>U</u> pdate an existing customer
DELETE	/api/customers/{customerId}	<u>D</u> delete an existing customer

Real-Time Project

Checkpoint

HTTP Method			CRUD Action
 POST	/api/customers		<u>C</u> reate a new customer
 GET	/api/customers		<u>R</u> ead a list of customers
 GET	/api/customers/{customerId}		<u>R</u> ead a single customer
PUT	/api/customers		<u>U</u> pdate an existing customer
DELETE	/api/customers/{customerId}		<u>D</u> delete an existing customer

Application Interaction

REST
Client

Customer
REST
Controller

Application Interaction

REST
Client

POST

/api/customers

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

Customer
REST
Controller

Application Interaction

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

REST
Client

POST

/api/customers

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

Customer
REST
Controller

Application Interaction

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

REST
Client

POST

/api/customers

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

Customer
REST
Controller

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


Application Interaction

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

REST
Client

POST

/api/customers

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

Customer
REST
Controller

Response contains
new id / primary key

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


Access the Request Body

Access the Request Body

- Jackson will convert request body from JSON to POJO

Access the Request Body

- Jackson will convert request body from JSON to POJO
- **@RequestBody** annotation binds the POJO to a method parameter

Access the Request Body

- Jackson will convert request body from JSON to POJO
- **@RequestBody** annotation binds the POJO to a method parameter

```
@PostMapping("/customers")  
public Customer addCustomer(@RequestBody Customer theCustomer) {  
  
    ...  
  
}
```


Access the Request Body

- Jackson will convert request body from JSON to POJO
- **@RequestBody** annotation binds the POJO to a method parameter

```
@PostMapping("/customers")  
public Customer addCustomer(@RequestBody Customer theCustomer) {  
  
    ...  
  
}
```

Now we can access the request body as a POJO

Add Customer

File: CustomerRestController.java

Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        ...

    }
}
```


Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        ...
    }
}
```

Use **@RequestBody** to access the request body as a POJO

Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```


Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```

Delegate call
to customer service

Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...


    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```



Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```

What's up with customer id?

What's up with customer id?

What's up with customer id?

- In the REST controller, we explicitly set the customer id to 0

What's up with customer id?

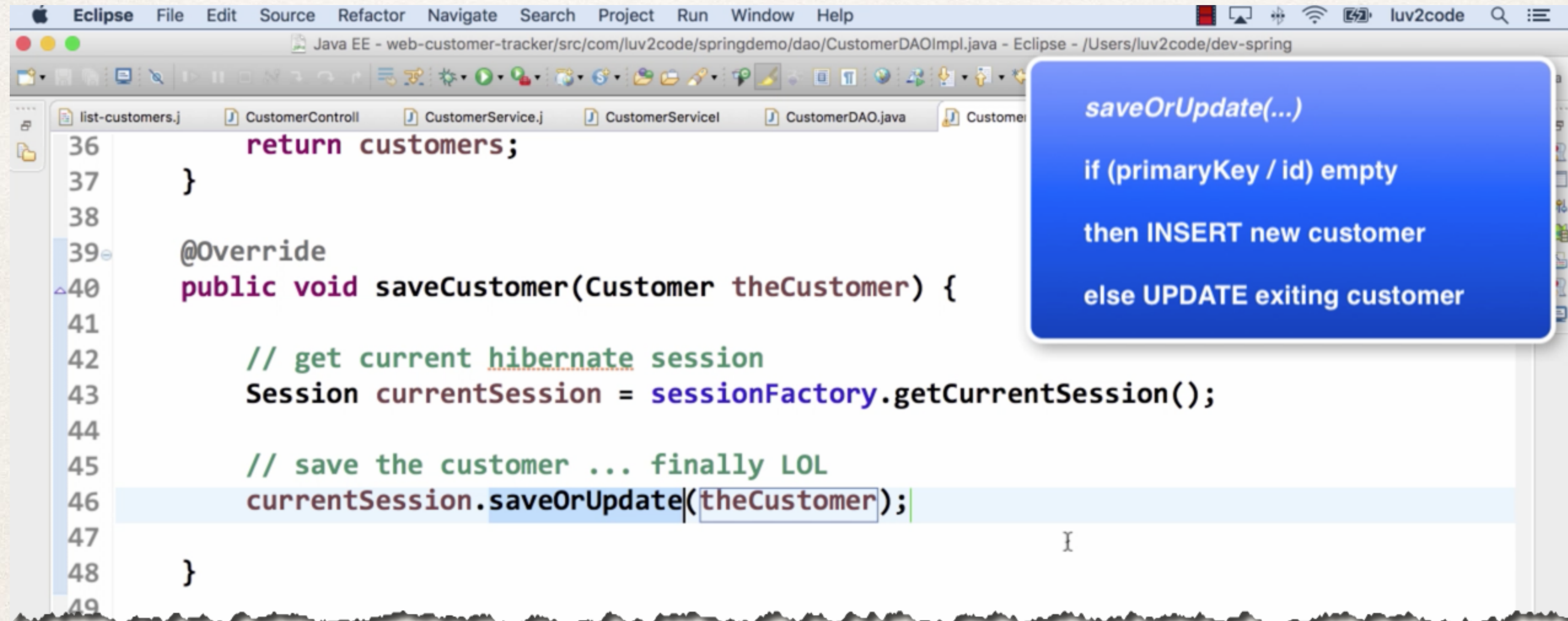
- In the REST controller, we explicitly set the customer id to 0
- Because our backend DAO code uses Hibernate method

What's up with customer id?

- In the REST controller, we explicitly set the customer id to 0
- Because our backend DAO code uses Hibernate method
 - `session.saveOrUpdate(...)`

Recall: CustomerDAOImpl

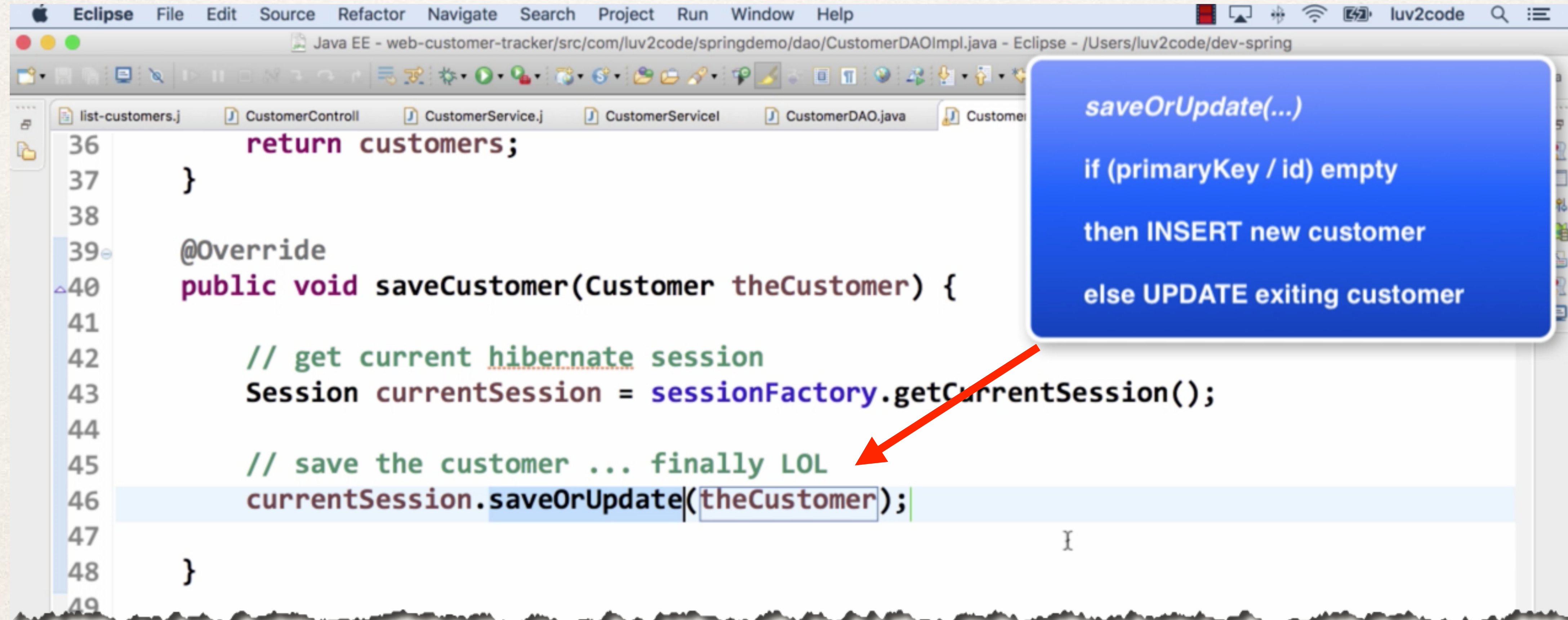
Recall: CustomerDAOImpl



```
36     return customers;
37 }
38
39 @Override
40 public void saveCustomer(Customer theCustomer) {
41
42     // get current hibernate session
43     Session currentSession = sessionFactory.getCurrentSession();
44
45     // save the customer ... finally LOL
46     currentSession.saveOrUpdate(theCustomer);
47
48 }
49
```

saveOrUpdate(...)
if (primaryKey / id) empty
then INSERT new customer
else UPDATE exiting customer

Recall: CustomerDAOImpl



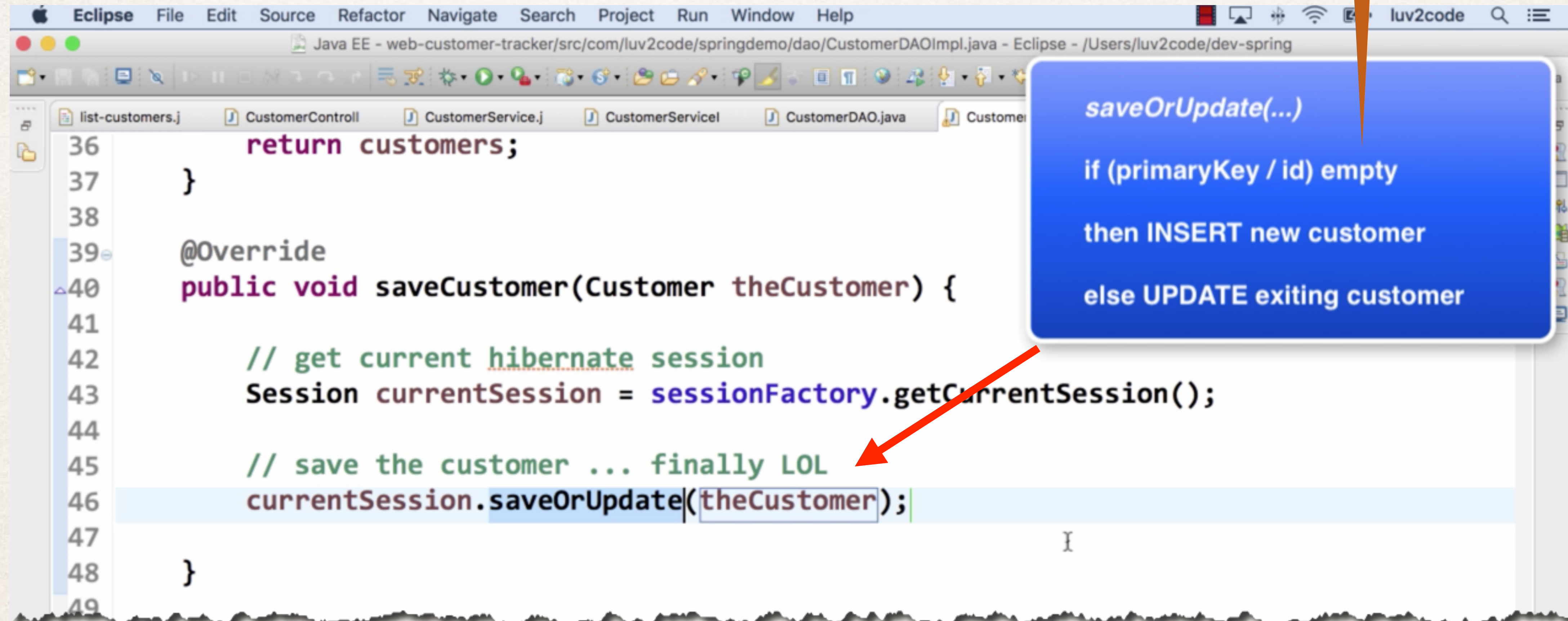
```
36     return customers;
37 }
38
39 @Override
40 public void saveCustomer(Customer theCustomer) {
41
42     // get current hibernate session
43     Session currentSession = sessionFactory.getCurrentSession();
44
45     // save the customer ... finally LOL
46     currentSession.saveOrUpdate(theCustomer);
47
48 }
49
```

saveOrUpdate(...)

- if (primaryKey / id) empty
- then INSERT new customer
- else UPDATE exiting customer

Recall: CustomerDAOImpl

Here: “empty” means
null or 0



```
36     return customers;
37 }
38
39 @Override
40 public void saveCustomer(Customer theCustomer) {
41
42     // get current hibernate session
43     Session currentSession = sessionFactory.getCurrentSession();
44
45     // save the customer ... finally LOL
46     currentSession.saveOrUpdate(theCustomer);
47
48 }
49
```

saveOrUpdate(...)
if (primaryKey / id) empty
then INSERT new customer
else UPDATE exiting customer

Adding customer with HTTP POST

Adding customer with HTTP POST

- If REST client is sending a request to “add”, using HTTP POST

Adding customer with HTTP POST

- If REST client is sending a request to “add”, using HTTP POST
- Then we ignore any id sent in the request

Adding customer with HTTP POST

- If REST client is sending a request to “add”, using HTTP POST
- Then we ignore any id sent in the request
- We overwrite the id with 0, to effectively set it to null / empty

Adding customer with HTTP POST

- If REST client is sending a request to “add”, using HTTP POST
- Then we ignore any id sent in the request
- We overwrite the id with 0, to effectively set it to null / empty
- Then our backend DAO code will “INSERT” new customer

Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```


Add Customer

File: CustomerRestController.java

```
@RestController
@RequestMapping("/api")
public class CustomerRestController {
    ...

    // add mapping for POST /customers - add new customer

    @PostMapping("/customers")
    public Customer addCustomer(@RequestBody Customer theCustomer) {

        theCustomer.setId(0);

        customerService.saveCustomer(theCustomer);

        return theCustomer;
    }
}
```

ID of 0 means
DAO code will
“INSERT” new customer

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

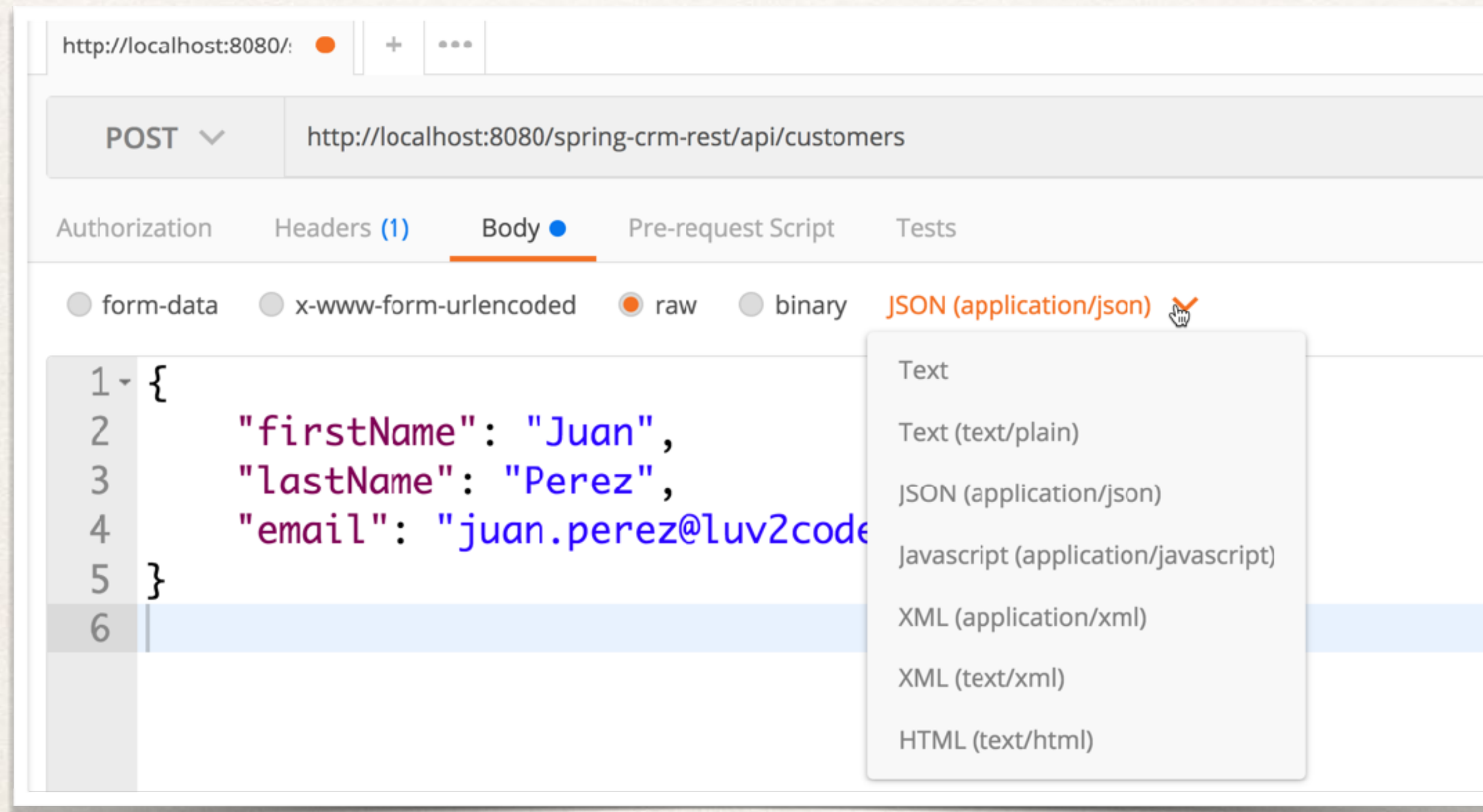
Postman - Sending JSON in Request Body

Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman

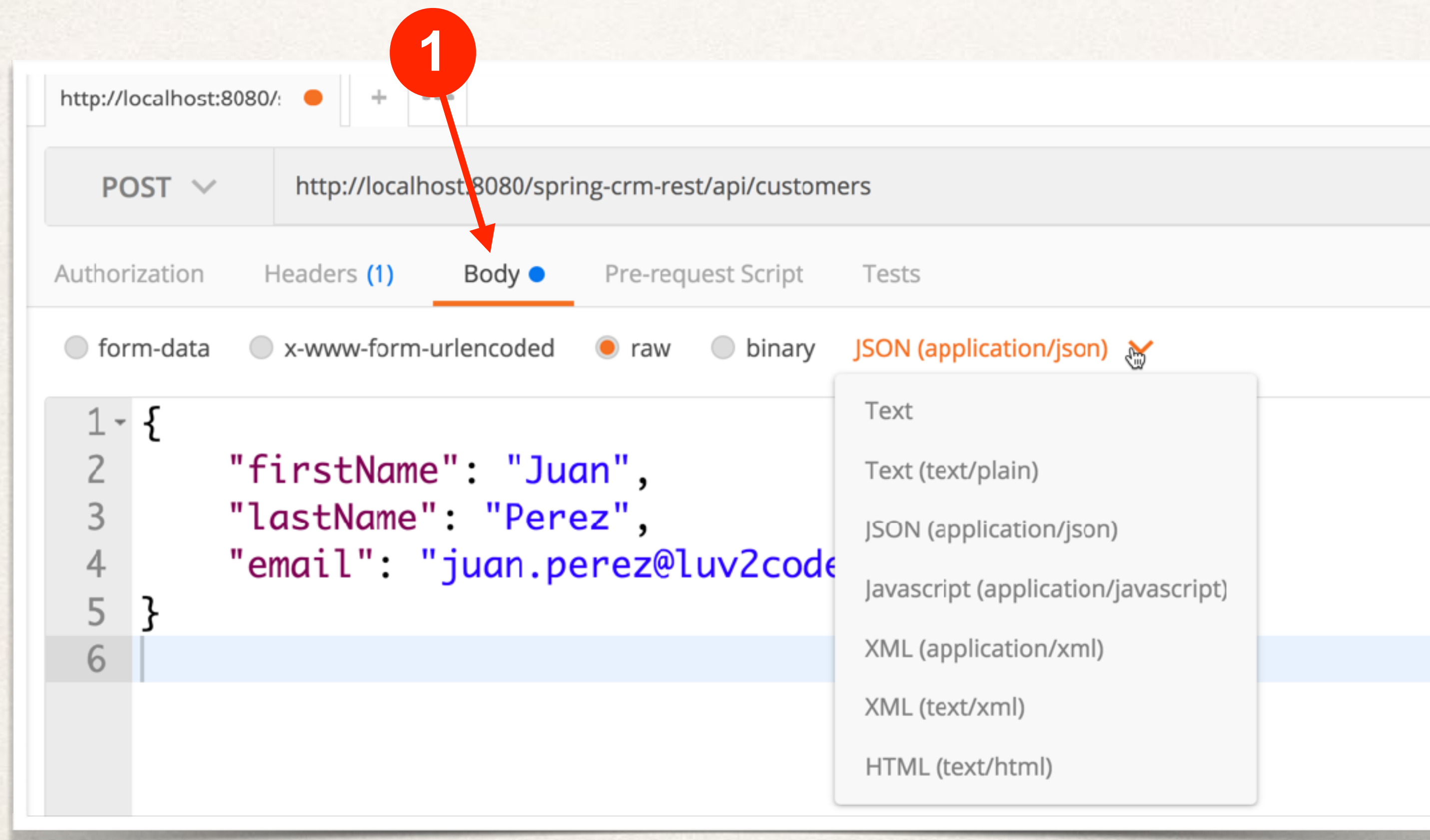
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



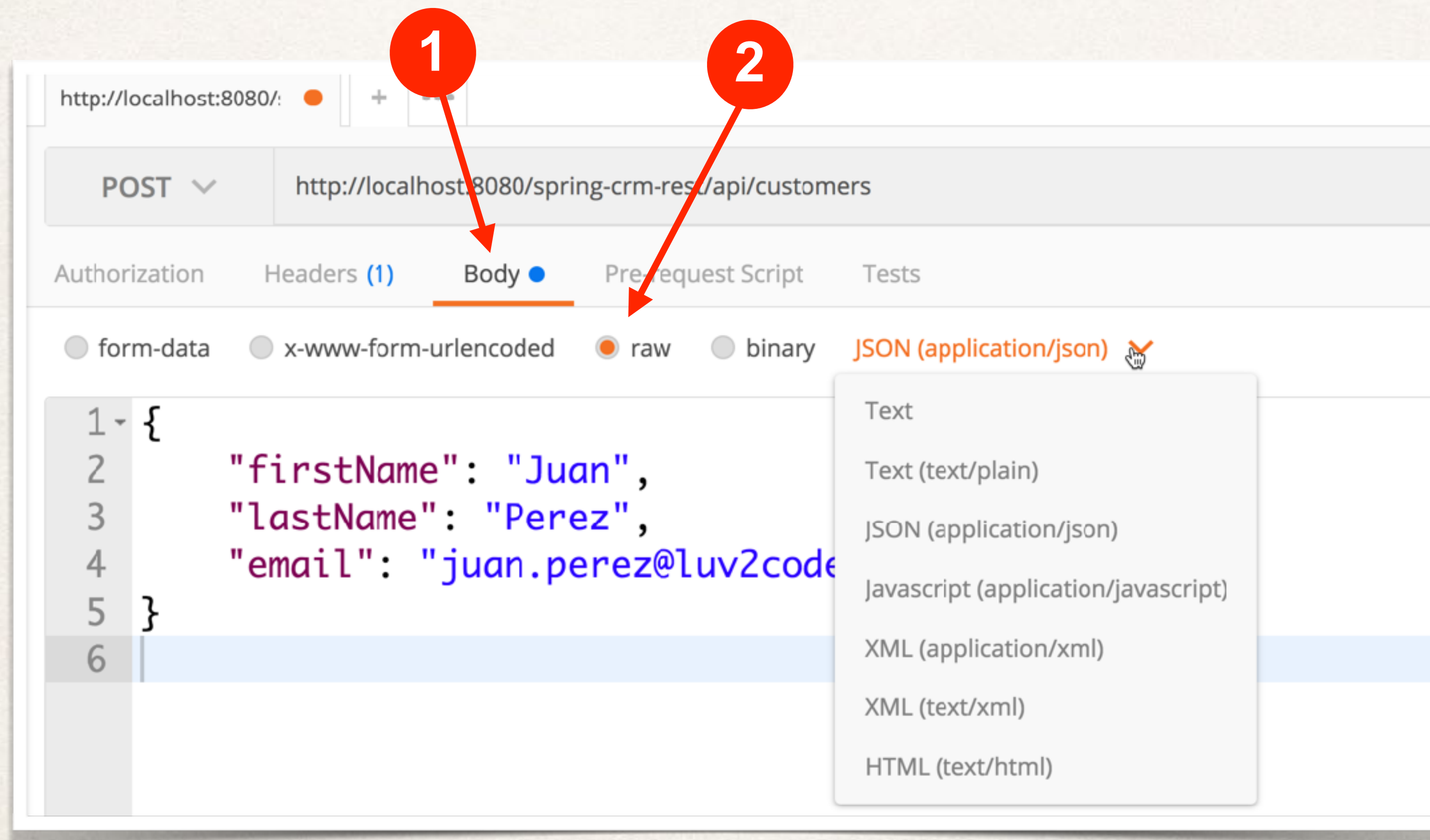
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



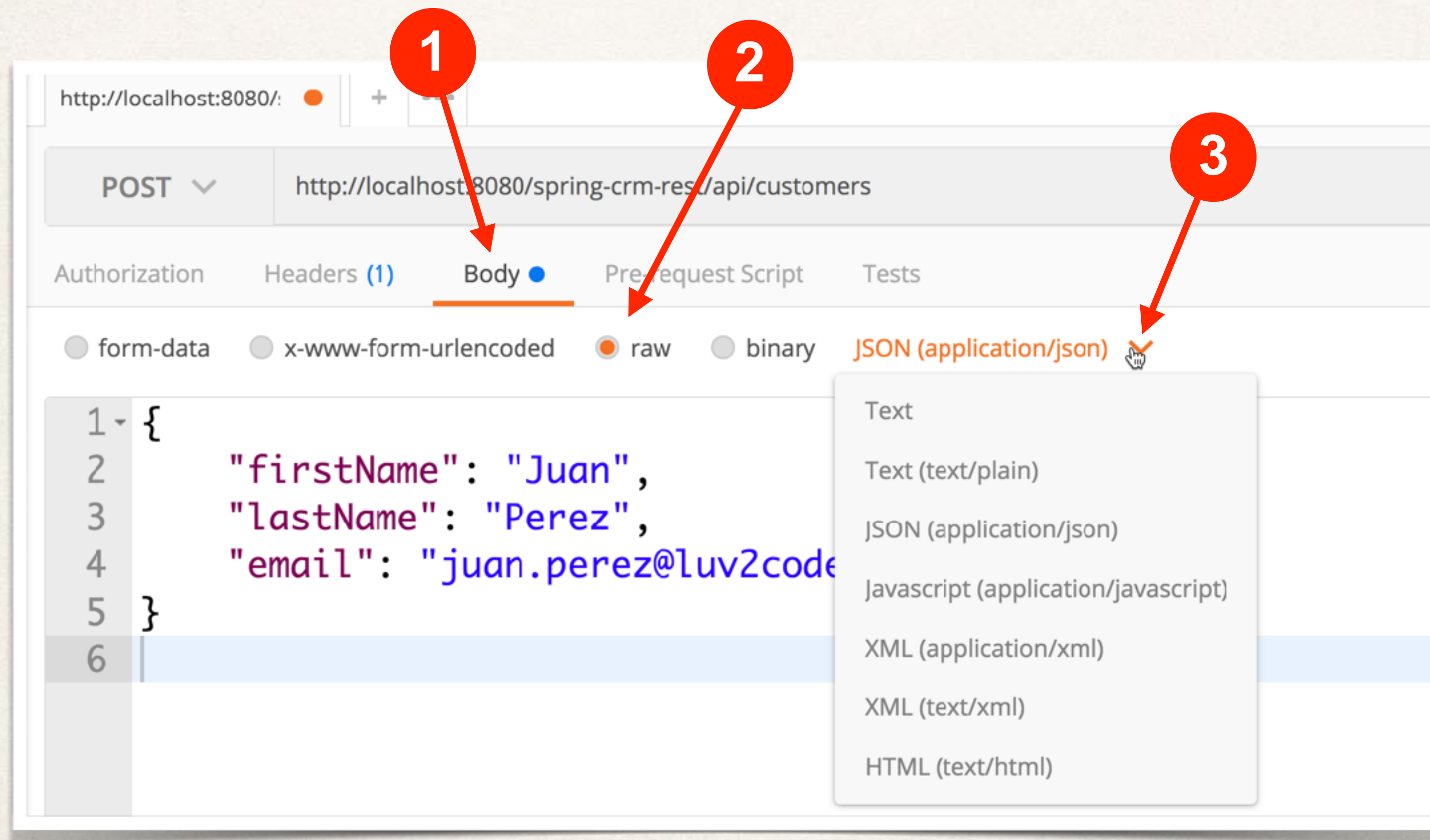
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



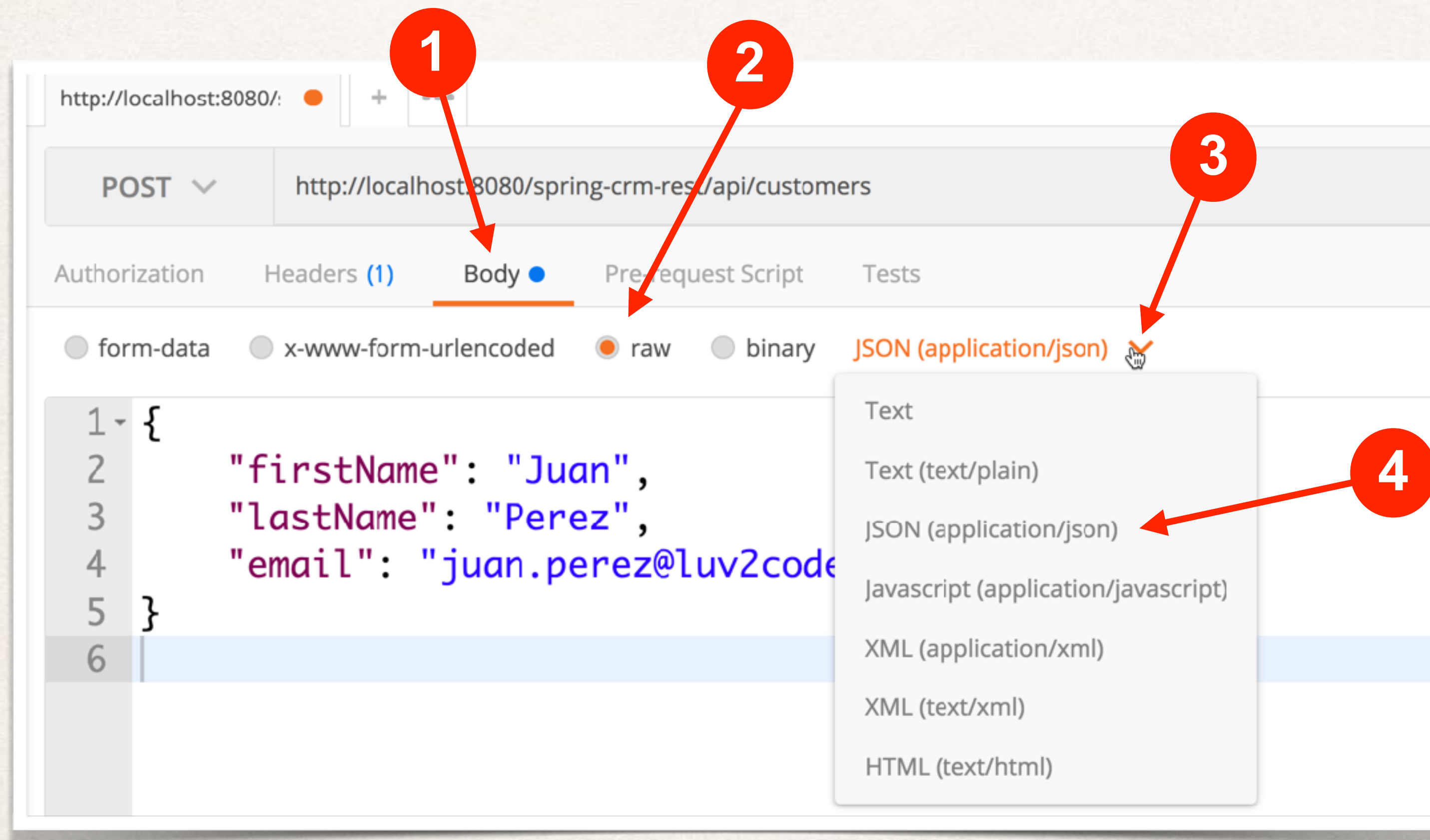
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



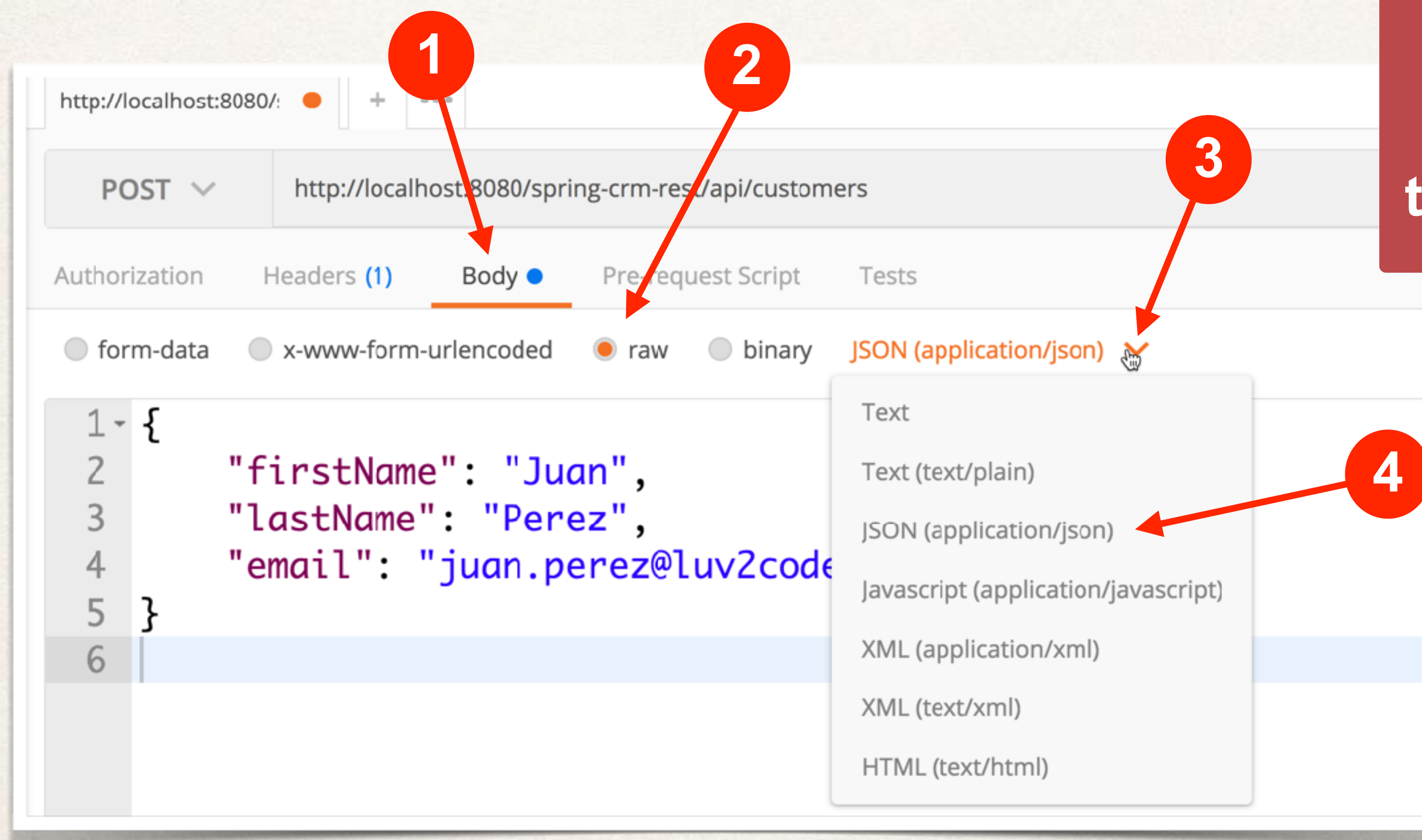
Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



Postman - Sending JSON in Request Body

- Must set HTTP request header in Postman



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