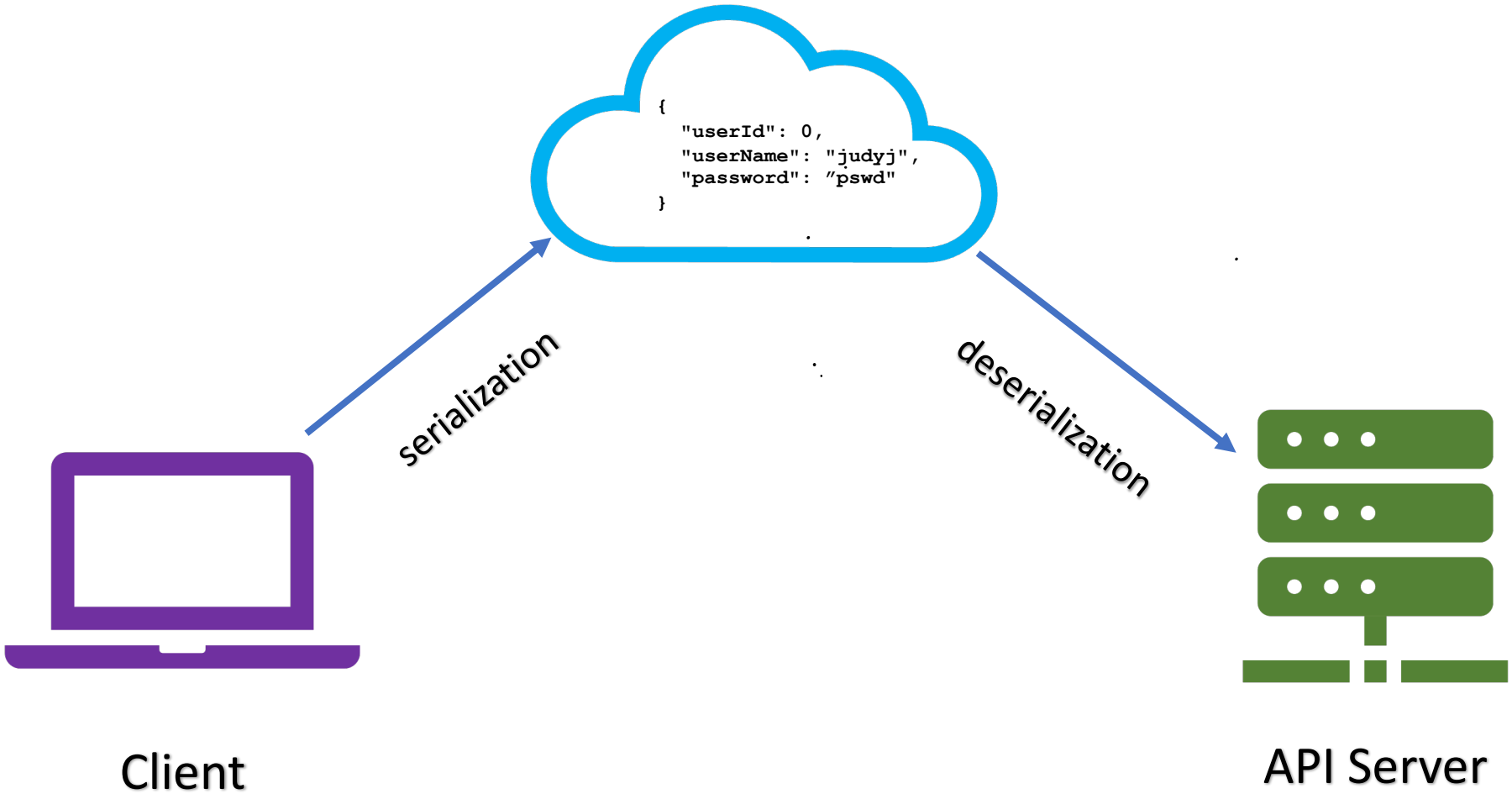
API Data Flow

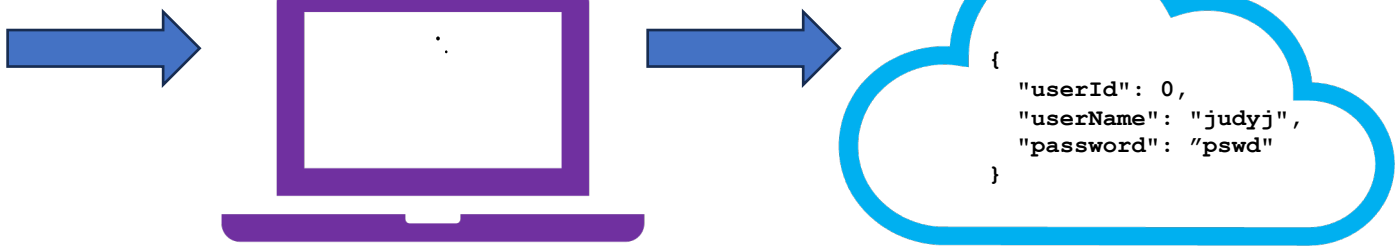


Data from Client

The `@JsonProperty` annotation says "send `pswd` as `password` in the JSON. "

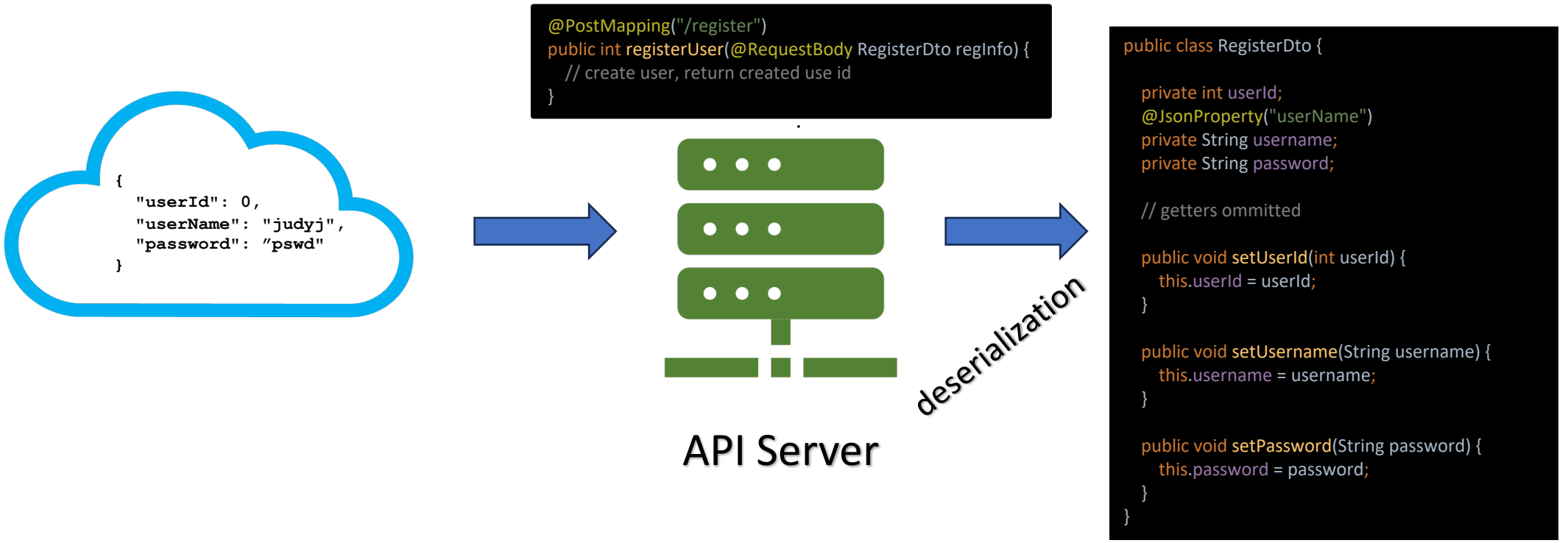
```
public class RegistrationInfo {

   private int userId;
   private String userName;
   @JsonProperty("password")
   private String pswd;

   // getters ommitted

   public void setUserId(int userId) {
      this.userId = userId;
   }

   public void setUserName(String userName) {
      this.userName = userName;
   }

   public void setPswd(String pswd) {
      this.pswd = pswd;
   }
}
```

```
restTemplate.postForObject(
// code to post RegistrationInfo object);
```



Client

Data to Server



Deserializtion Process

The `@JsonProperty` annotation says “when you see **`userName`** in the JSON, it neans **`username`** in this class”

Using standard setter notation, setters are called based on property name (i.e. **`userId`** calls **`setUserId`**)

Behind the scenes (what Spring is doing):

- new RegisterDto()
- setUserId(0)
- setUsername(”judyj”)
- setPassword(“p`swd”)

```java
public class RegisterDto {

    private int userId;
    @JsonProperty("userName")
    private String username;
    private String password;

    // getters ommitted

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Note that **`userName`** in the JSON would map to **`setUserName`**, in the Java class, which does not exist. Which would mean that the **`username`** property would ***not*** get set.

`@JsonProperty` resolves the mismatch