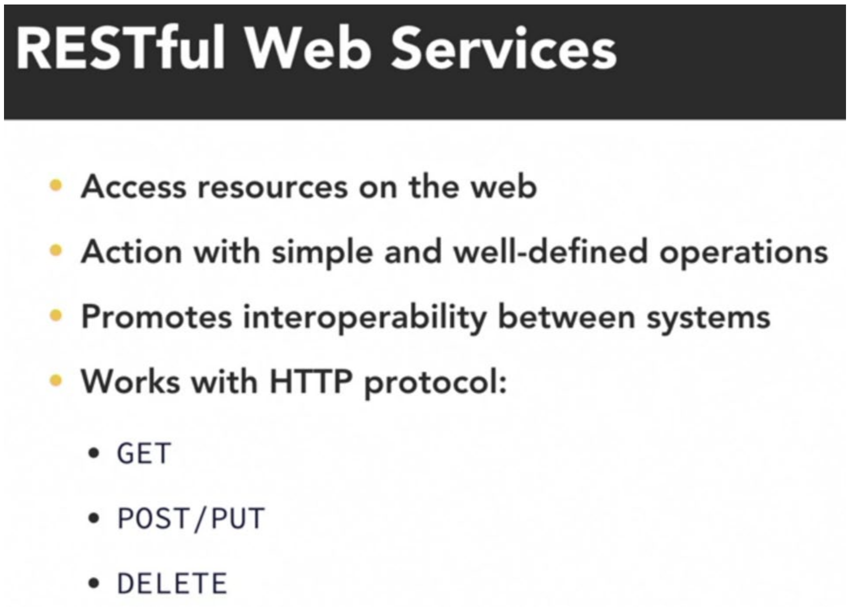
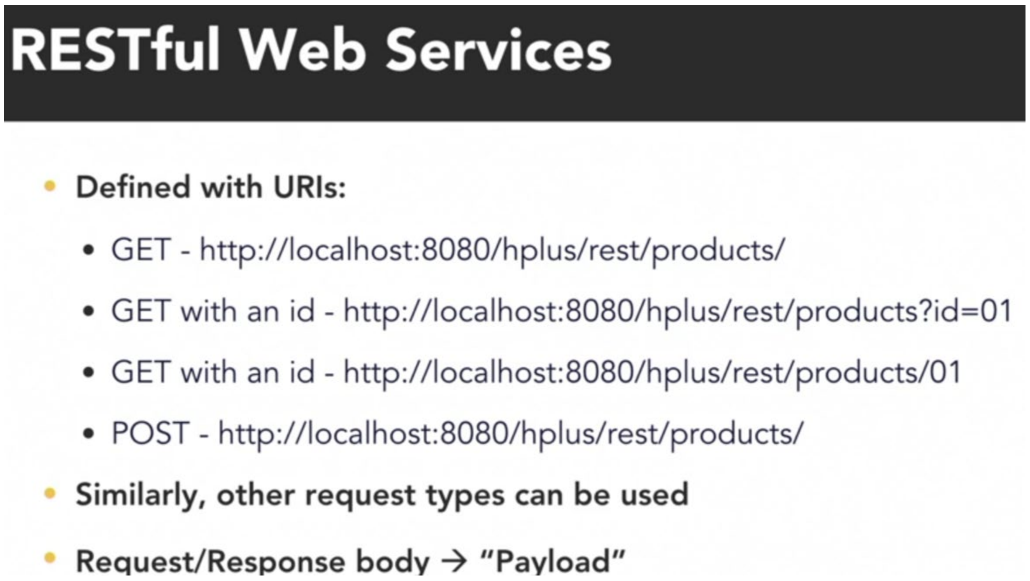
Spring MVC Rest Controller



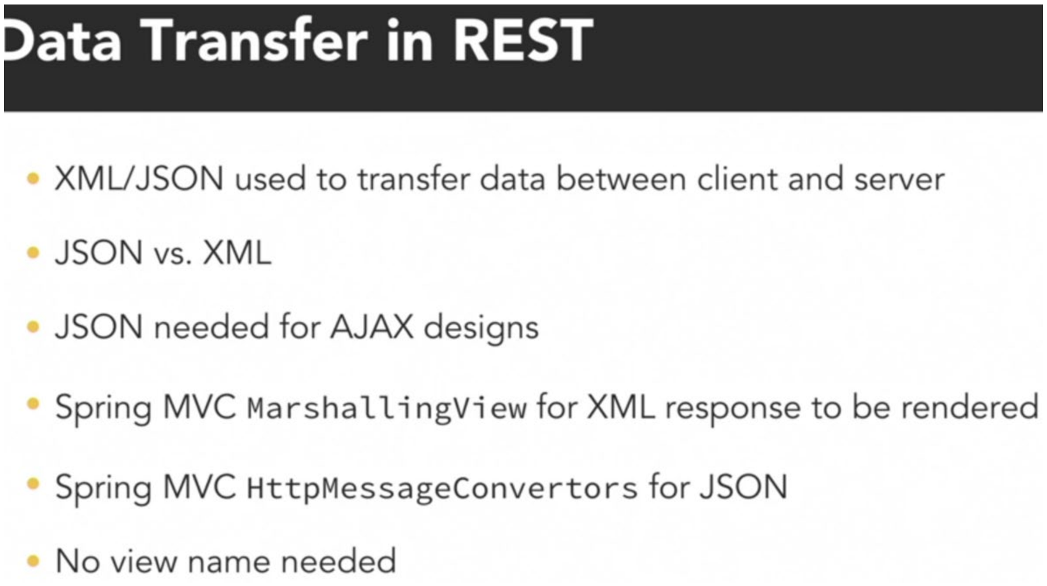
RESTful Web Services

- Access resources on the web
- Action with simple and well-defined operations
- Promotes interoperability between systems
- Works with HTTP protocol:
  - GET
  - POST/PUT
  - DELETE



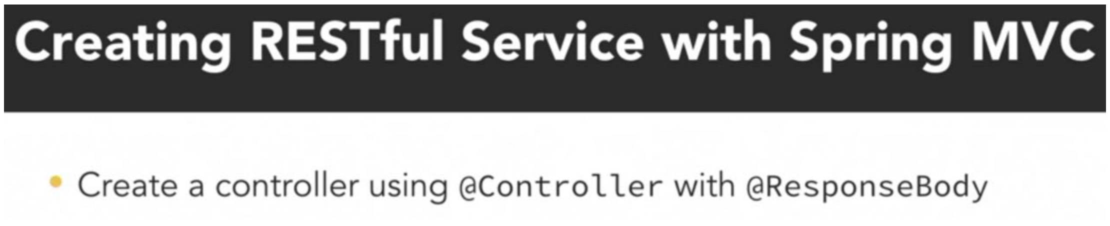
RESTful Web Services

- Defined with URIs:
  - GET - http://localhost:8080/hplus/rest/products/
  - GET with an id - http://localhost:8080/hplus/rest/products?id=01
  - GET with an id - http://localhost:8080/hplus/rest/products/01
  - POST - http://localhost:8080/hplus/rest/products/
- Similarly, other request types can be used
- Request/Response body → "Payload"



Data Transfer in REST

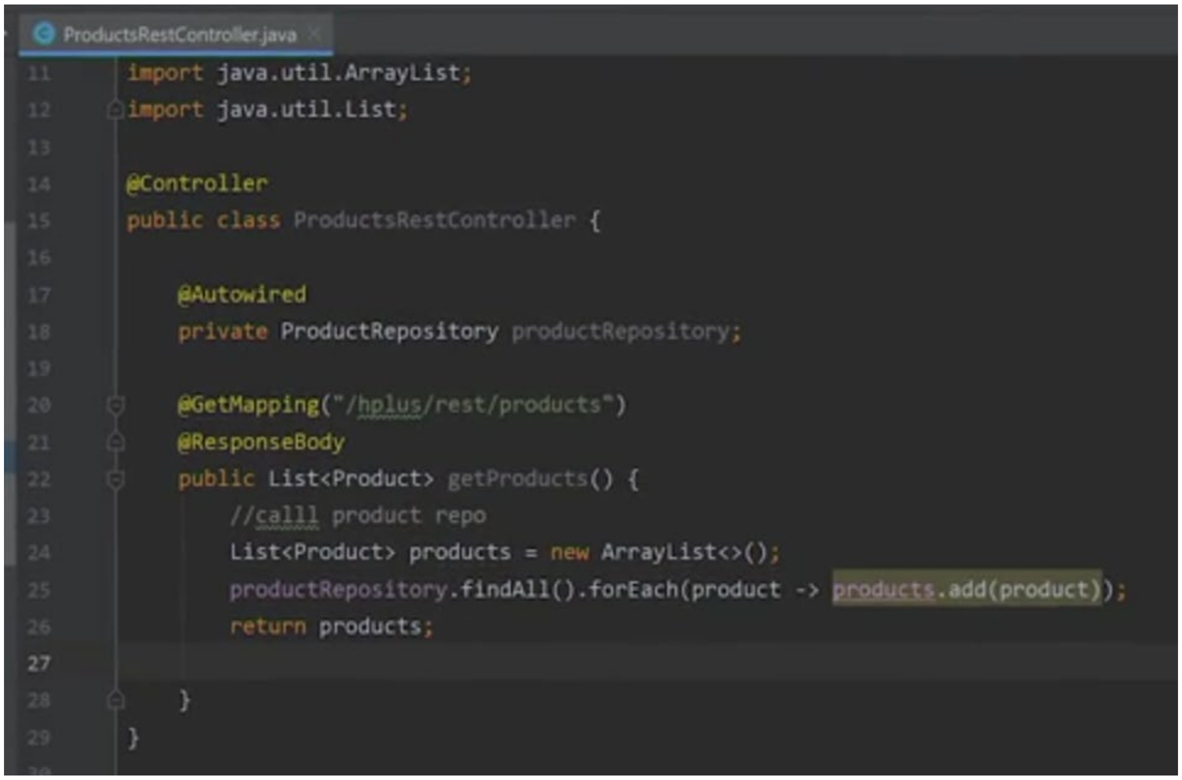
- XML/JSON used to transfer data between client and server
- JSON vs. XML
- JSON needed for AJAX designs
- Spring MVC `MarshallingView` for XML response to be rendered
- Spring MVC `HttpMessageConvertors` for JSON
- No view name needed



Creating RESTful Service with Spring MVC

- Create a controller using `@Controller` with `@ResponseBody`

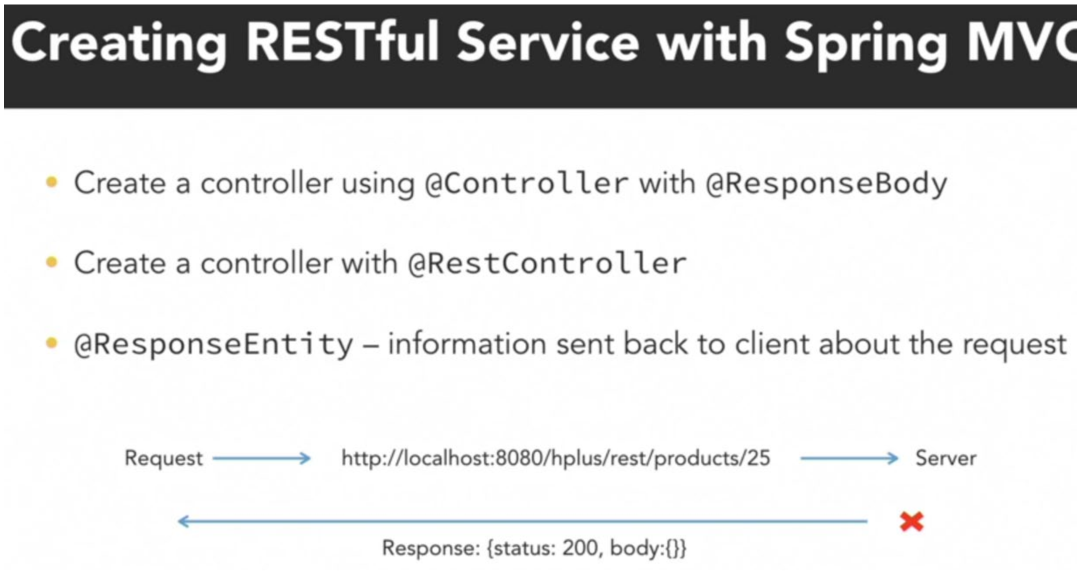
Create a Rest Controller called ProductsRestController



```java
import java.util.ArrayList;
import java.util.List;

@Controller
public class ProductsRestController {

    @Autowired
    private ProductRepository productRepository;

    @GetMapping("/hplus/rest/products")
    @ResponseBody
    public List<Product> getProducts() {
        //calll product repo
        List<Product> products = new ArrayList<>();
        productRepository.findAll().forEach(product -> products.add(product));
        return products;

    }
}
```

Now build the application and access the url

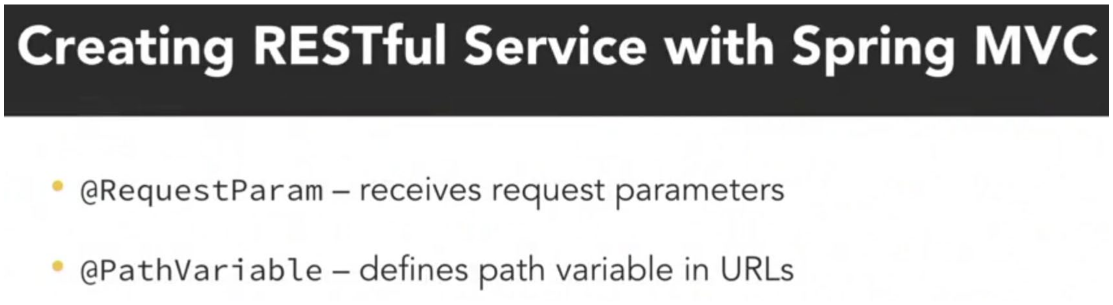
http://localhost:8080//hplus/rest/products



Creating RESTful Service with Spring MVC

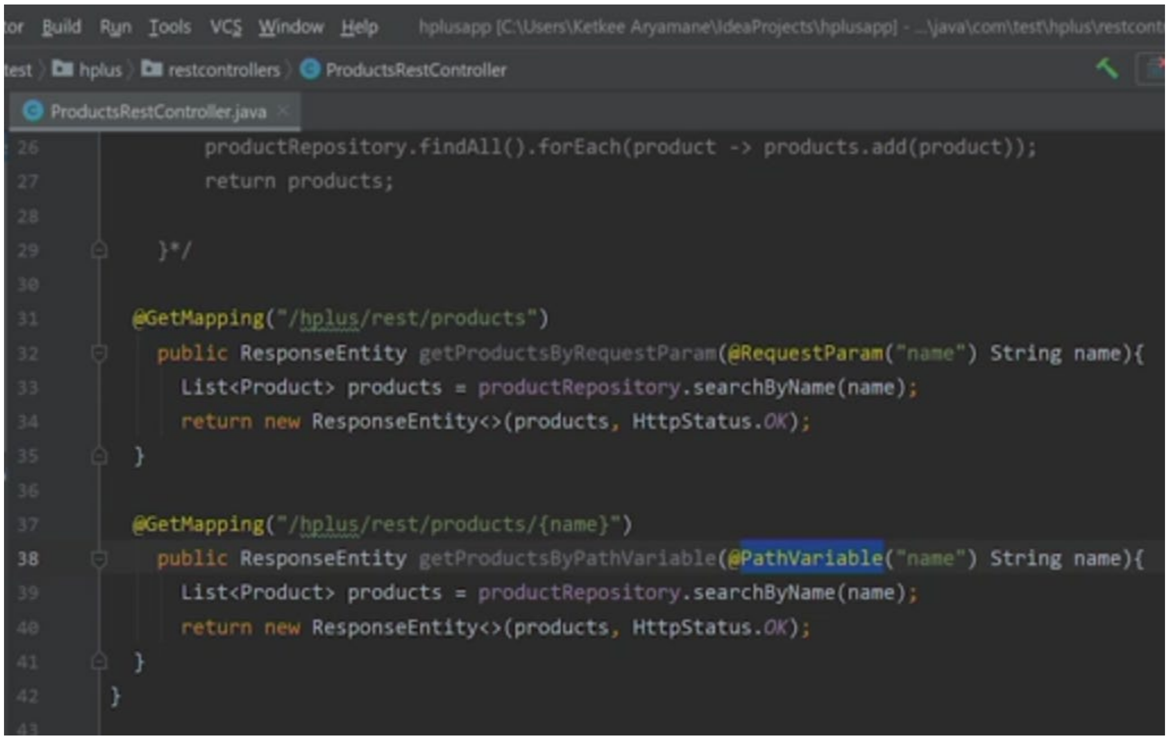
- Create a controller using `@Controller` with `@ResponseBody`
- Create a controller with `@RestController`
- `@ResponseEntity` – information sent back to client about the request

Request
http://localhost:8080/hplus/rest/products/25
Server
Response: {status: 200, body:{}}



Creating RESTful Service with Spring MVC

- `@RequestParam` – receives request parameters
- `@PathVariable` – defines path variable in URLs


tor Build Run Tools VCS Window Help hplusapp [C:\Users\Ketkee Aryamane\IdeaProjects\hplusapp] - ...\java\com\test\hplus\restcont
test hplus restcontrollers ProductsRestController
ProductsRestController.java

```
        productRepository.findAll().forEach(product -> products.add(product));
        return products;

    }*/

  @GetMapping("/hplus/rest/products")
    public ResponseEntity getProductsByRequestParam(@RequestParam("name") String name){
      List<Product> products = productRepository.searchByName(name);
      return new ResponseEntity<>(products, HttpStatus.OK);
  }

  @GetMapping("/hplus/rest/products/{name}")
    public ResponseEntity getProductsByPathVariable(@PathVariable("name") String name){
      List<Product> products = productRepository.searchByName(name);
      return new ResponseEntity<>(products, HttpStatus.OK);
  }
}
```

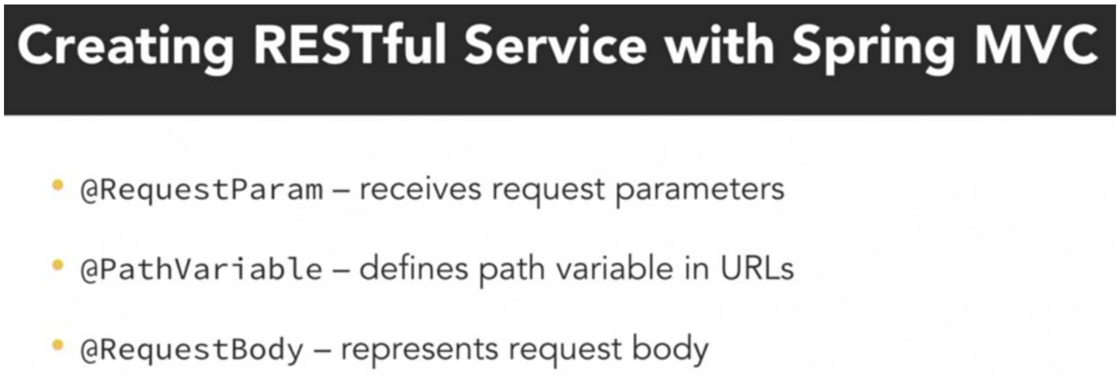
Now build the application and access the url

As Param

http://localhost:8080//hplus/rest/products?name=water

And as path variable

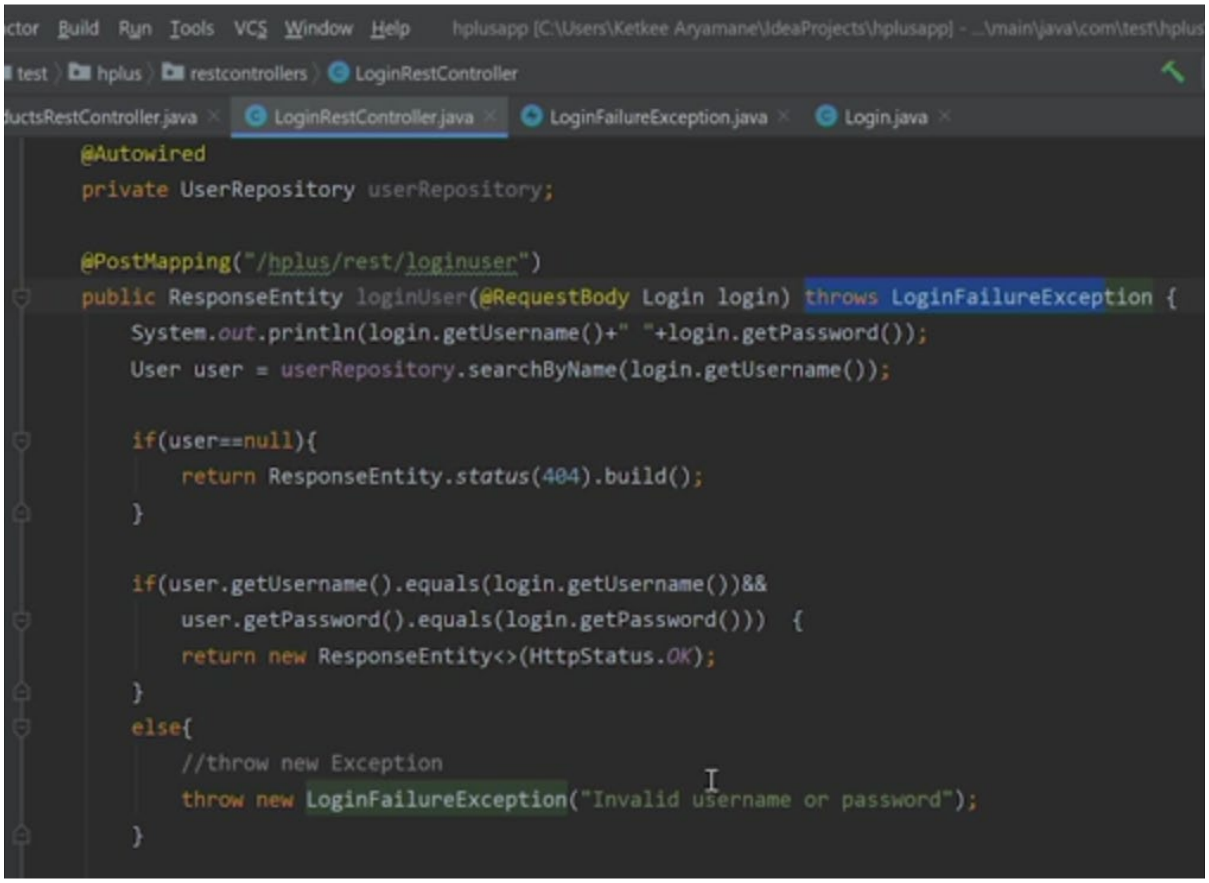
http://localhost:8080//hplus/rest/products/water



Creating RESTful Service with Spring MVC

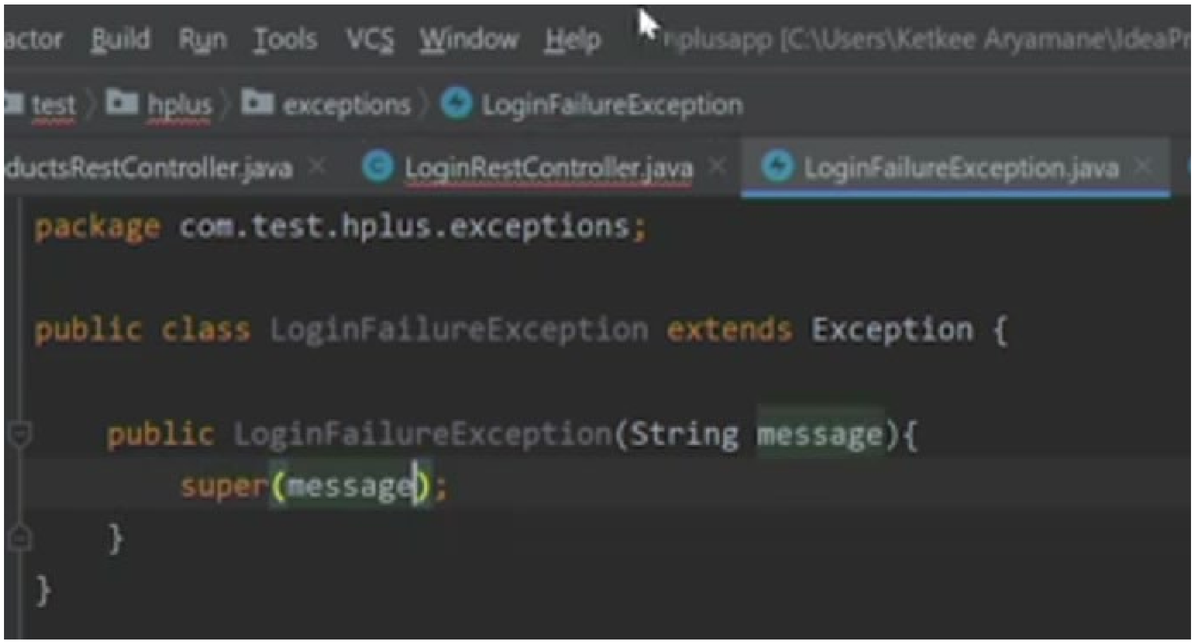
- @RequestParam – receives request parameters
- @PathVariable – defines path variable in URLs
- @RequestBody – represents request body

Create a LoginRestController

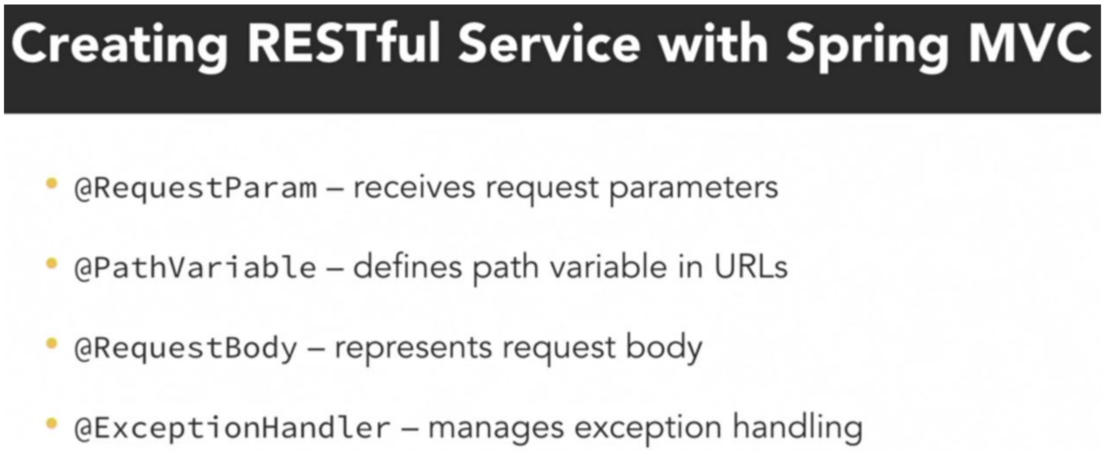

ctor Build Run Tools VCS Window Help hplusapp [C:\Users\Ketkee Aryamane\IdeaProjects\hplusapp] - ...\main\java\com\test\hplus
test hplus restcontrollers LoginRestController
ductsRestController.java LoginRestController.java LoginFailureException.java Login.java

```java
    @Autowired
    private UserRepository userRepository;

    @PostMapping("/hplus/rest/loginuser")
    public ResponseEntity loginUser(@RequestBody Login login) throws LoginFailureException {
        System.out.println(login.getUsername()+" "+login.getPassword());
        User user = userRepository.searchByName(login.getUsername());

        if(user==null){
            return ResponseEntity.status(404).build();
        }

        if(user.getUsername().equals(login.getUsername())&&
            user.getPassword().equals(login.getPassword()))  {
            return new ResponseEntity<>(HttpStatus.OK);
        }
        else{
            //throw new Exception
            throw new LoginFailureException("Invalid username or password");
        }
```

Create a LoginFailureException to throw error when user password is wrong


actor Build Run Tools VCS Window Help hplusapp [C:\Users\Ketkee Aryamane\IdeaPr
test hplus exceptions LoginFailureException
ductsRestController.java LoginRestController.java LoginFailureException.java

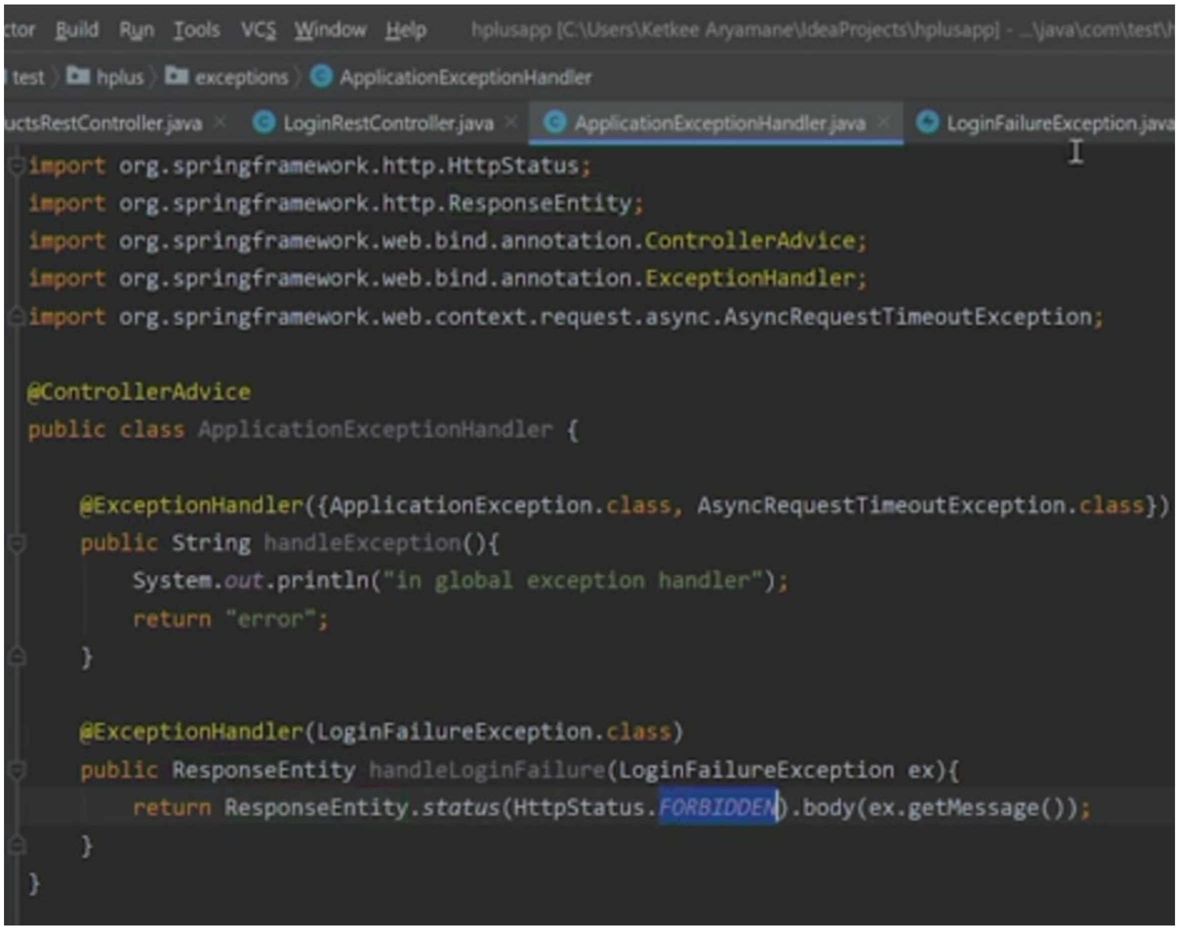
```java
package com.test.hplus.exceptions;

public class LoginFailureException extends Exception {

    public LoginFailureException(String message){
        super(message);
    }
}
```



Creating RESTful Service with Spring MVC

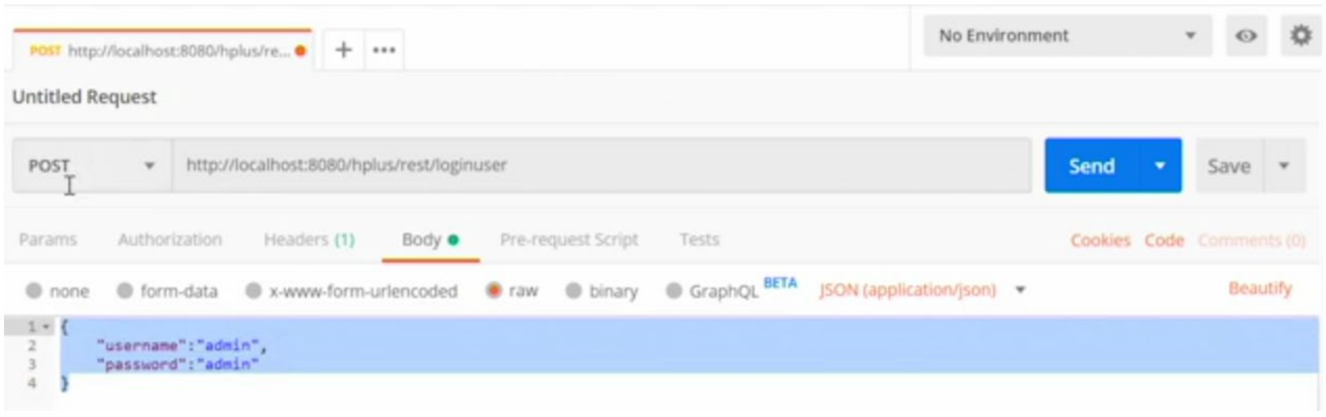
- `@RequestParam` – receives request parameters
- `@PathVariable` – defines path variable in URLs
- `@RequestBody` – represents request body
- `@ExceptionHandler` – manages exception handling

Inside ApplicationExceptionHandler add one more Handler for handling the LoginFailureException.


ctor Build Run Tools VCS Window Help hplusapp [C:\Users\Ketkee Aryamane\IdeaProjects\hplusapp] - ...\java\com\test\
test hplus exceptions ApplicationExceptionHandler
uctsRestController.java LoginRestController.java ApplicationExceptionHandler.java LoginFailureException.java

```java
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.async.AsyncRequestTimeoutException;

@ControllerAdvice
public class ApplicationExceptionHandler {

    @ExceptionHandler({ApplicationException.class, AsyncRequestTimeoutException.class})
    public String handleException(){
        System.out.println("in global exception handler");
        return "error";
    }

    @ExceptionHandler(LoginFailureException.class)
    public ResponseEntity handleLoginFailure(LoginFailureException ex){
        return ResponseEntity.status(HttpStatus.FORBIDDEN).body(ex.getMessage());
    }
}
```

Access

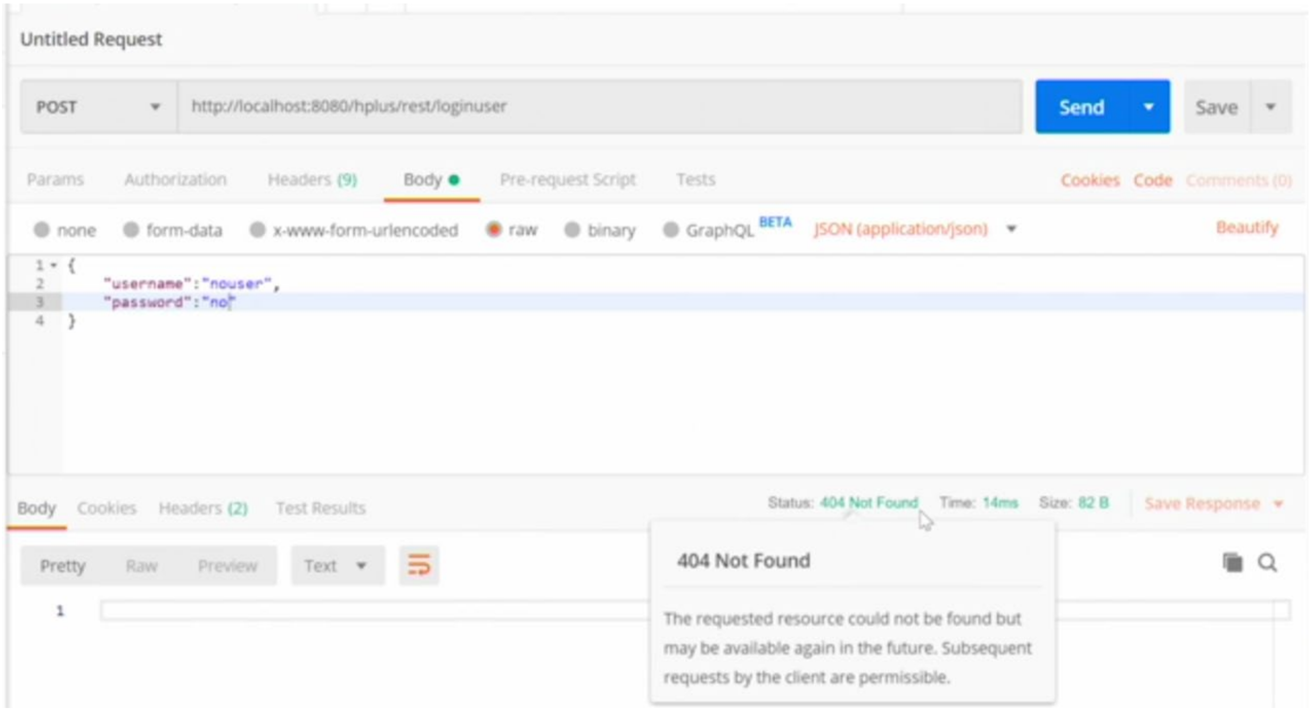


Untitled Request

POST http://localhost:8080/hplus/rest/loginuser

Body: raw, JSON (application/json)

```
{
    "username":"admin",
    "password":"admin"
}
```



Untitled Request

POST http://localhost:8080/hplus/rest/loginuser

Body: raw, JSON (application/json)

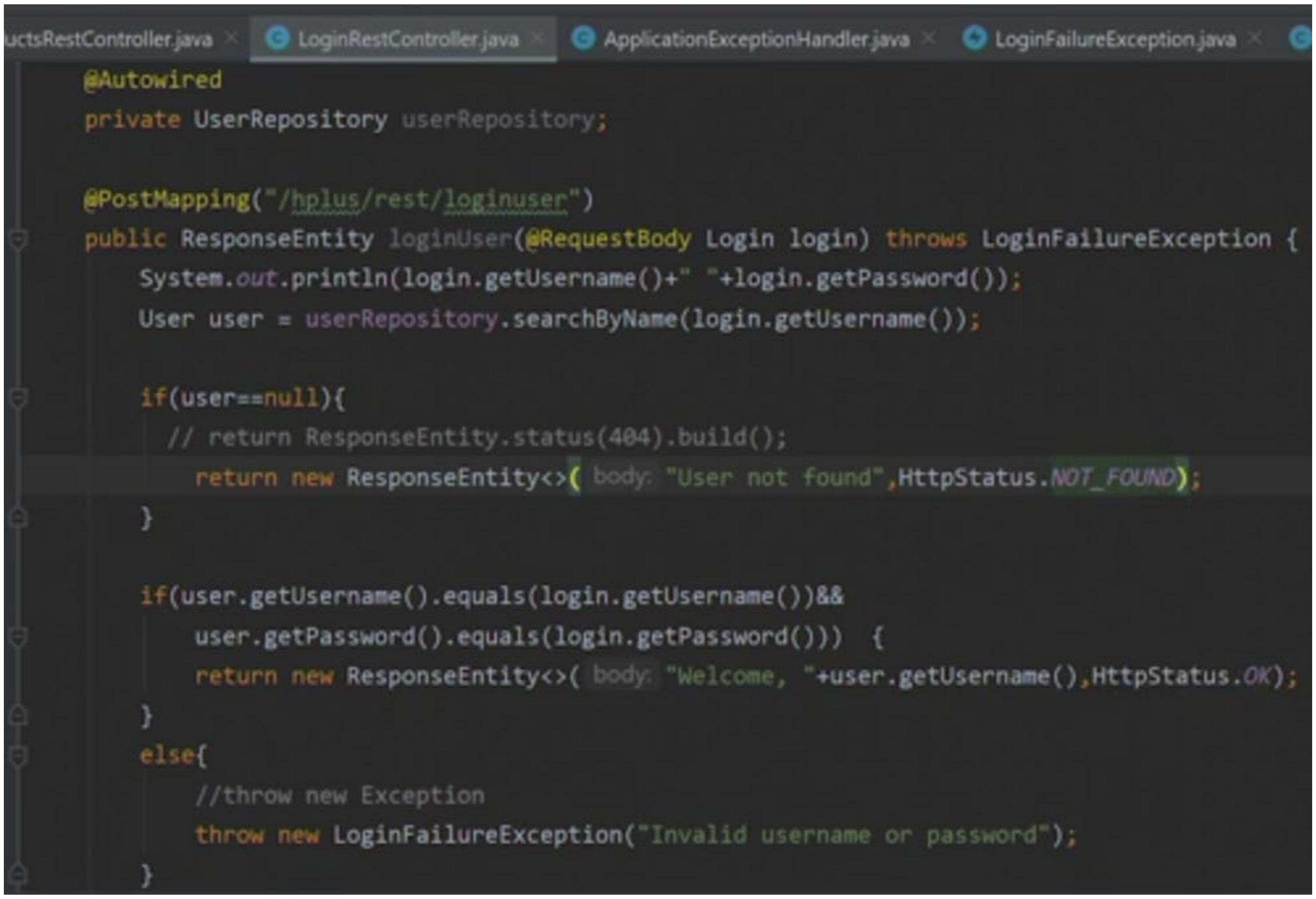
```
{
    "username":"nouser",
    "password":"no"
}
```

Status: 404 Not Found Time: 14ms Size: 82 B

404 Not Found

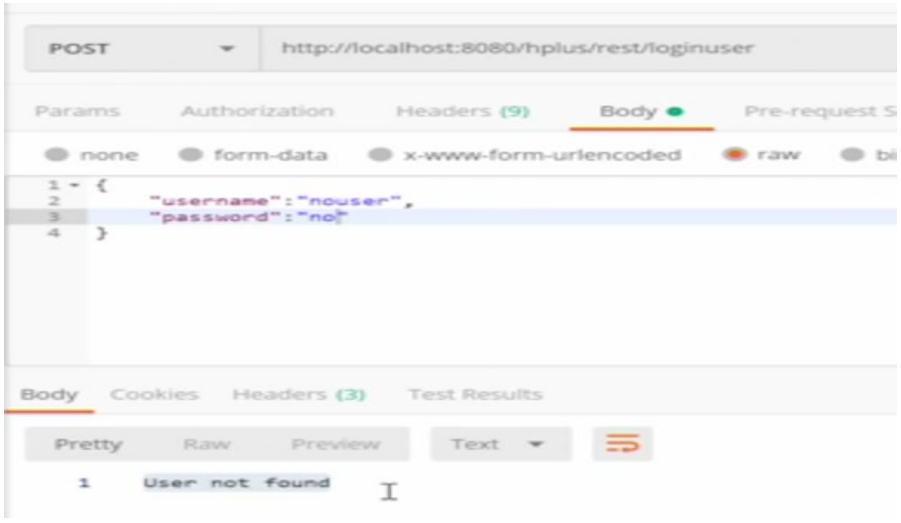
The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

Add More info



```
    @Autowired
    private UserRepository userRepository;

    @PostMapping("/hplus/rest/loginuser")
    public ResponseEntity loginUser(@RequestBody Login login) throws LoginFailureException {
        System.out.println(login.getUsername()+" "+login.getPassword());
        User user = userRepository.searchByName(login.getUsername());

        if(user==null){
         // return ResponseEntity.status(404).build();
            return new ResponseEntity<>( body: "User not found",HttpStatus.NOT_FOUND);
        }

        if(user.getUsername().equals(login.getUsername())&&
            user.getPassword().equals(login.getPassword()))  {
            return new ResponseEntity<>( body: "Welcome, "+user.getUsername(),HttpStatus.OK);
        }
        else{
            //throw new Exception
            throw new LoginFailureException("Invalid username or password");
        }
```

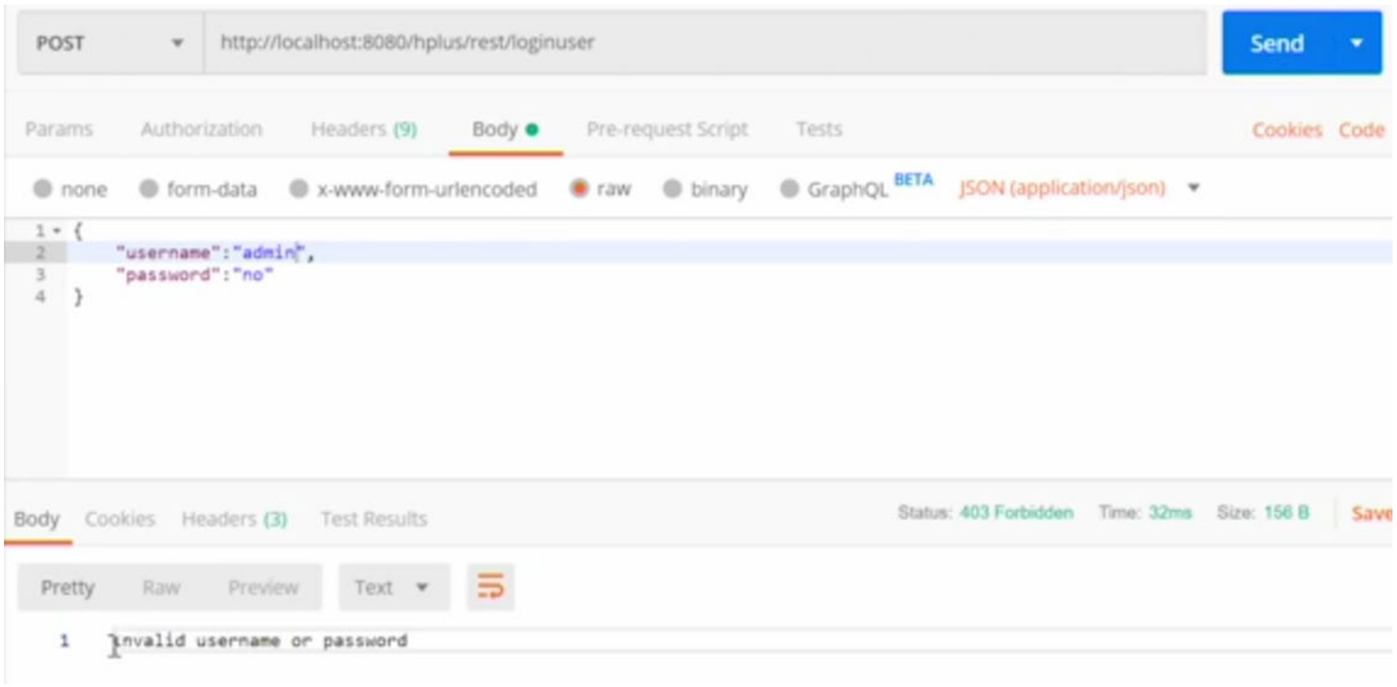
Access



```
http://localhost:8080/hplus/rest/loginuser
```

```
{
    "username":"nouser",
    "password":"no"
}
```

```
User not found
```



POST http://localhost:8080/hplus/rest/loginuser

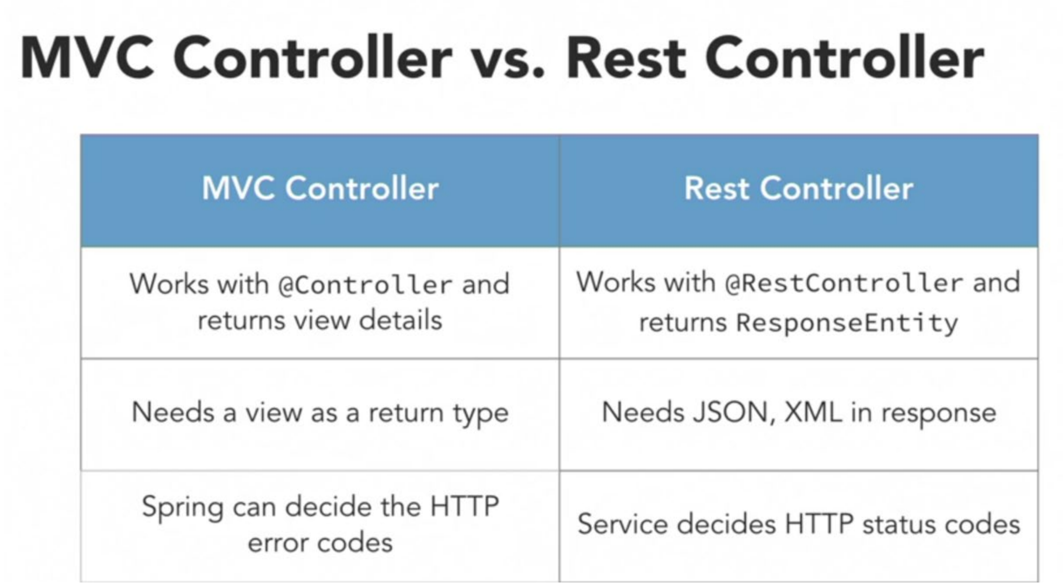
Body: raw, JSON (application/json)

```
{
    "username":"admin",
    "password":"no"
}
```

Status: 403 Forbidden Time: 32ms Size: 156 B

Text

```
invalid username or password
```



MVC Controller vs. Rest Controller

MVC Controller	Rest Controller
Works with `@Controller` and returns view details	Works with `@RestController` and returns `ResponseEntity`
Needs a view as a return type	Needs JSON, XML in response
Spring can decide the HTTP error codes	Service decides HTTP status codes