Day 13

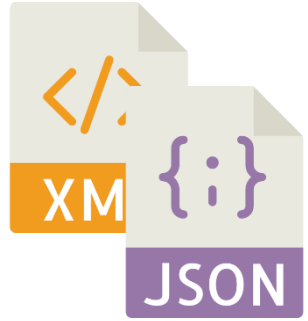
Sending Data to Web Application

- Data can be send from the client to the web application
- Use HTTP POST method to send data to the web application
  - Data/payload is transport in the body of the request
    - After the last HTTP header
  - HTTP header `Content-Type` specifies the payload formt/encoding
- POST is used to create/insert data
  - Eg a new order entry, where the payload is the detail of the order

Form

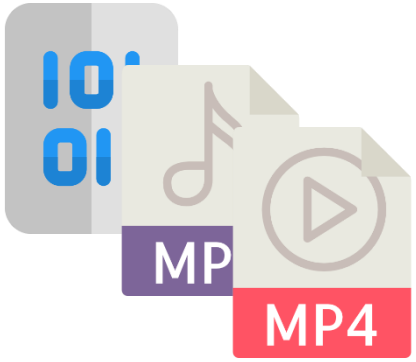
`application/x-www-form-urlencoded`

Structured data



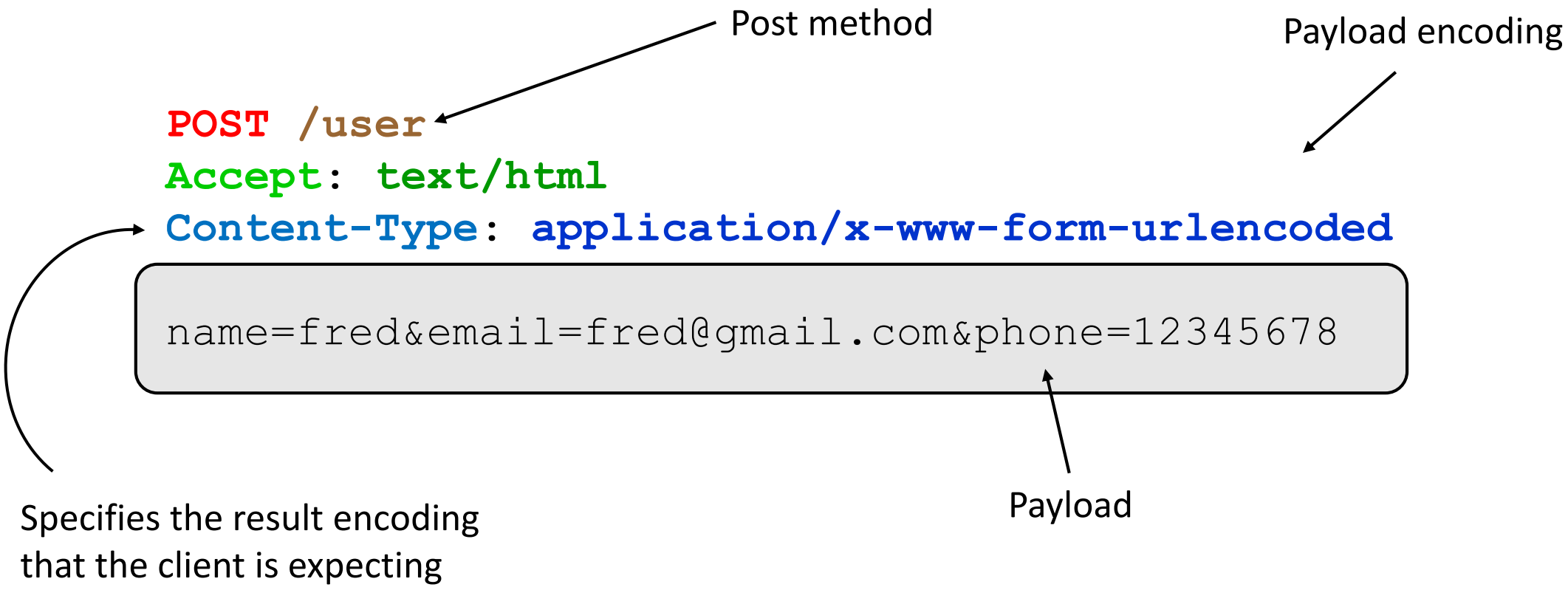
`application/json, application/xml`

Large binary file



`multipart/form-data`

HTTP POST


Post method
Payload encoding
POST /user
Accept: text/html
Content-Type: application/x-www-form-urlencoded
name=fred&email=fred@gmail.com&phone=12345678
Specifies the result encoding that the client is expecting
Payload

Handling a POST Request

```
@Controller
@RequestMapping(path="/user")
public class UserController {

   @PostMapping(
      consumes = "application/x-www-form-urlencoded",
      produces = "text/html")
   public String createUser(
      @ModelAttribute User user, Model model) {

      // process the data
   }
}
```

POST /user should be processed by this method

For matching `Accepts` and `Content-Type`

Form fields are mapped to this object

Processing HTTP POST Request - 1

```
<form method="POST" action="/user">
    <input type="text" name="name">
    <input type="email" name="email">
    <input type="tel" name="phone">
    <button type="submit">
</form>
```

Form field are encoded in `x-www-form-urlencoded`

```
POST /user
Accept: text/html
Content-Type: application/x-www-form-urlencoded

name=fred&email=fred@gmail.com&phone=12345678
```

Processing HTTP POST Request - 2

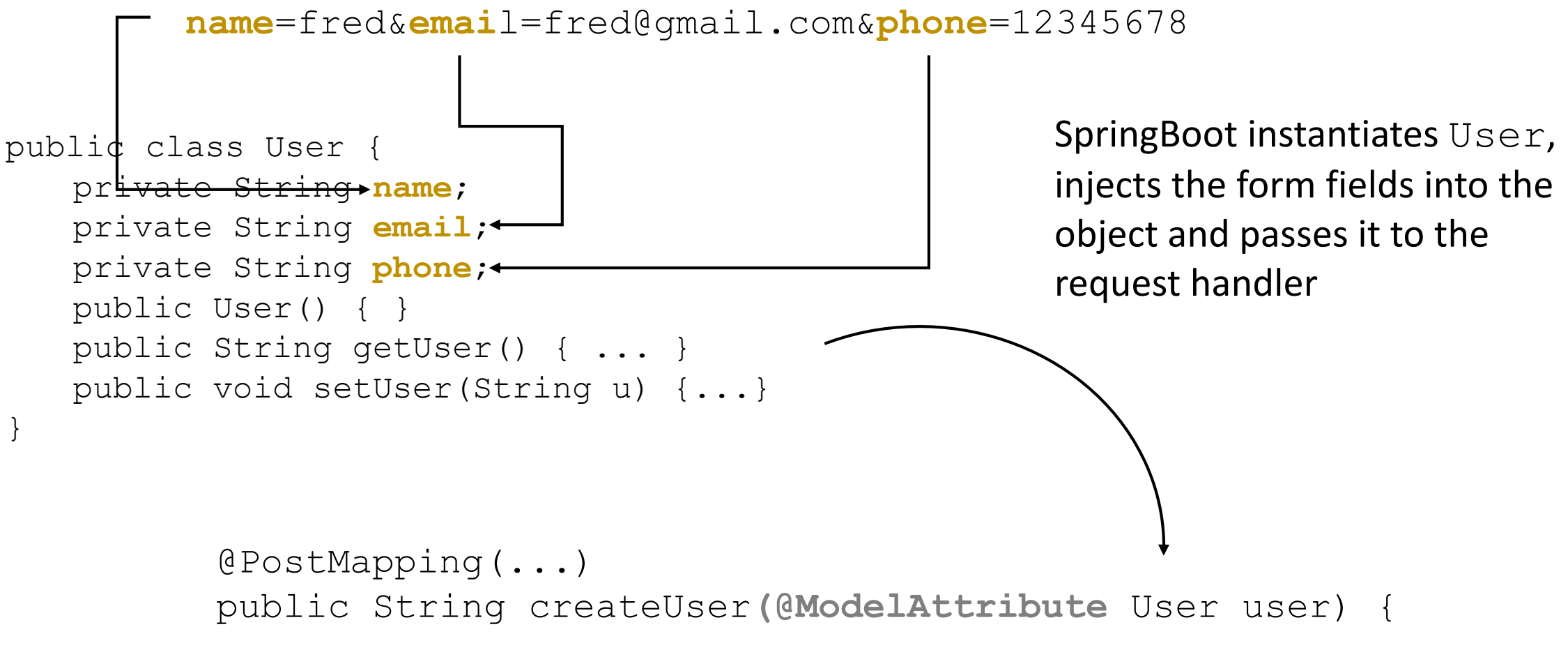
```
POST /user
Accept: text/html
Content-Type: application/x-www-form-urlencoded

name=fred&email=fred@gmail.com&phone=12345678
```

```
@Controller
@RequestMapping(path="/user")
public class UserController {
    @PostMapping(
        consumes = "application/x-www-form-urlencoded",
        produces = "text/html")
    public String createUser(@ModelAttribute User user, Model model) {
        // process the data
        ...
    }
}
```

User object is created from form fields

Mapping Form Fields - @ModelAttribute



SpringBoot instantiates `User`, injects the form fields into the object and passes it to the request handler

Mapping Form Fields - `MultiValueMap`

```
name=fred&email=fred@gmail.com&phone=12345678
```

Use `@RequestBody` to map payload to `MultiValueMap`

All form fields are added to the Map
More appropriate if there are lots of inputs from the from

```
@PostMapping(...)
public String createUser(
    @RequestBody MultiValueMap<String, String> form,
    Model model) {

        String name = form.getFirst("name");
        String email = form.getFirst("email");
        String phone = form.getFirst("phone");
        ...
}
```

Difference Between `GET` and `POST`

GET

- Bookmarkable because the parameters are part of the URL
- Limited to 255 characters
- Results are cached

POST

- Data are carried in the body
- Need to specify the media type with `Content-Type` header
- No payload size limit

- Typical use in form submission
- GET - To retrieve some data, eg. searching for a book
- POST - To create some data, eg. RSVP a wedding