

01_SpringBoot实现图片上传

时间：2023年02月10日 14:40:06

方案一：将图片使用base64编码为字符串

请求中携带：

- filename
- base64编码的图片数据

```
@RestController
@RequestMapping("/images")
public class ImageController {

    0 个用法 新 *
    @PostMapping("/upload")
    public void multiImageImport(@RequestBody ImageUploadDTO imageUploadInfo) {

        try (BufferedOutputStream bo = new BufferedOutputStream(new FileOutputStream( name: "backup-" + imageUploadInfo
            .getFilename())) {
            bo.write(Base64Utils.decodeFromString(imageUploadInfo.getImage()));
            bo.flush();
        } catch (IOException e) {
            // todo log
            throw new RuntimeException(e);
        }
    }
}
```

```
@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
@AllArgsConstructor
public class ImageUploadDTO {

    0 个用法
    private String filename;

    // base64 编码的单个图片
    0 个用法
    private String image;
}
```

方案二：使用MultipartFile

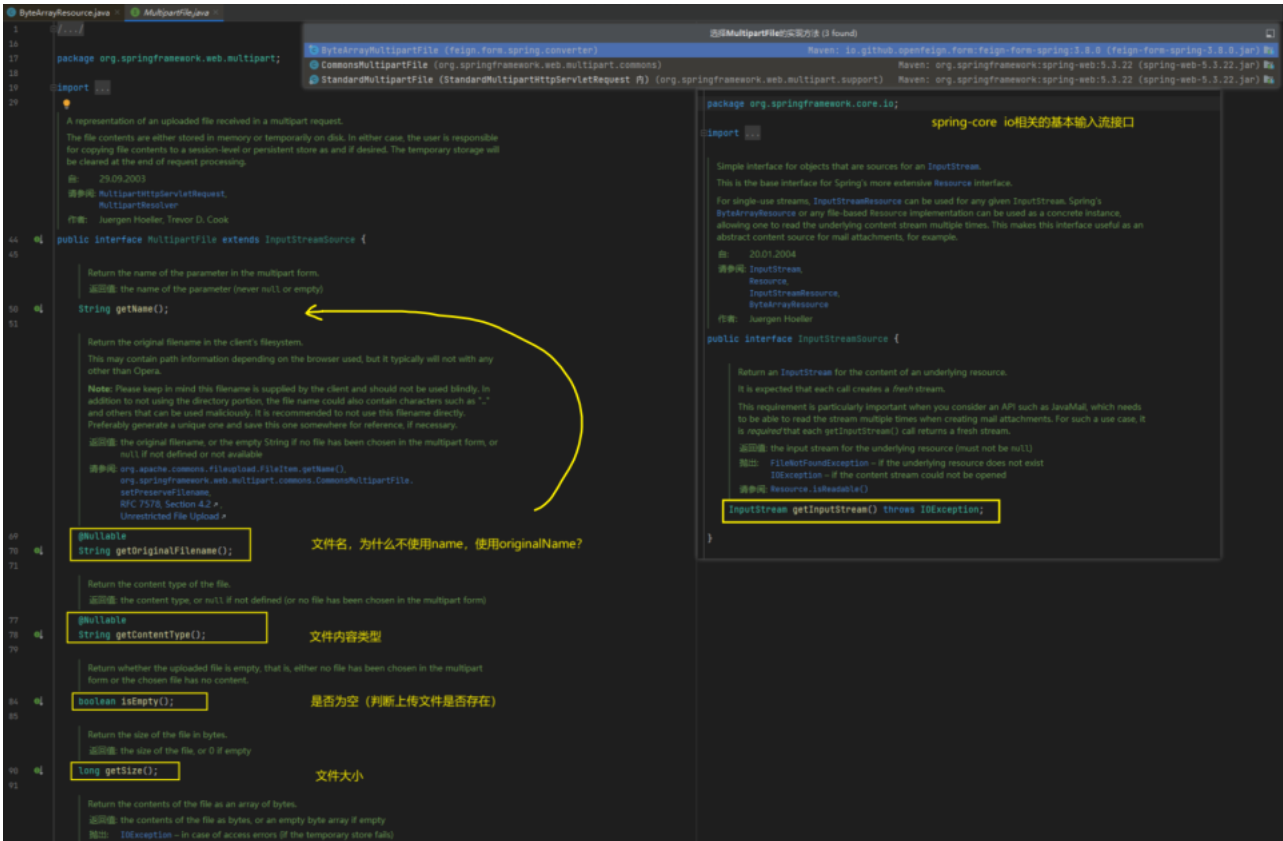
HTML中有没有一种统一的文件上传方式，例如通过表单的encrypt类型指定。

- enctype 属性规定在将表单数据发送到服务器之前如何对其进行编码。

值	描述
application/x-www-form-urlencoded	默认。在发送前对所有字符进行编码（将空格转换为 "+" 符号，特殊字符转换为 ASCII HEX 值）。
multipart/form-data	不对字符编码。当使用有文件上传控件的表单时，该值是必需的。
text/plain	将空格转换为 "+" 符号，但不编码特殊字符。

1. 什么是MultipartFile

- html表单中的一种数据encrypt格式，用于上传文件
- spring中提供的一个接口+多个实现，用于接收并处理前端web表单请求中的文件数据



- **getName**
 - 获取请求文件的属性名

POST localhost:9091/images/upload

参数 授权 Header (9) Body 预请求脚本 测试 设置

none form-data x-www-form-urlencoded raw binary GraphQL

键	值	描述
<input checked="" type="checkbox"/> file	yjtp.png ×	
键	值	描述

-
- **getOriginalName**
 - 获取文件名
- **getContentType**
 - image/png
 - ...(MIME类型)
- **getSize**
 - 获取文件大小 (字节)
- **isEmpty**
 - 上传文件是否为空 -> getSize==0 (例如一个空的.txt文件)

2. 多文件使用Multipart接收

- 不使用注解指定文件所在的key
- 使用注解指定文件所在的key，此时可以使用@RequestParam或@RequestParam。

```
/**
 * 根据表单属性解析出Multipart，可以使用@RequestParam、也可以使用@RequestParam
 *
 * @param fileArray 多文件
 */
0 个用法 新 *
@PostMapping("/upload")
public void multiImageImport(@RequestParam("files") MultipartFile[] fileArray) {

    System.out.println(Arrays.toString(fileArray));
}
```

```
0 个用法 新 *
@PostMapping("/upload")
public void multiImageImport(MultipartFile[] file) {
    System.out.println("file = " + file);
}
```

多文件直接使用数组接收

估表达式(Enter)或添加监视(Ctrl+Shift+Enter)

```

this = {ImageController@6556}
file = {MultipartFile[2]@6558}
0 = {StandardMultipartHttpServletRequest$StandardMultipartFile@6585}
  > part = {ApplicationPart@6587}
  > filename = "新文件 1.json"
1 = {StandardMultipartHttpServletRequest$StandardMultipartFile@6586}
  > part = {ApplicationPart@6589}
  > filename = "新文件 2.json"
```

3. Multipart数据写入文件

- 使用MultipartFile提供的transferTo (File)，但是这里只支持绝对路径，因为相对路径默认不在项目目录下！

```

/**
 * 根据表单属性解析出Multipart，可以使用@RequestPart、也可以使用@RequestParam
 *
 * @param fileArray 多文件
 */
0 个用法 新 *
@PostMapping("/upload")
public void multiImageImport(@RequestPart("files") MultipartFile[] fileArray) {

    for (MultipartFile multipartFile : fileArray) {
        String filename = Objects.requireNonNull(multipartFile.getOriginalFilename());
        try {
            OutputStream os = new FileOutputStream(filename);
            InputStream is = multipartFile.getInputStream();

            multipartFile.transferTo(new File(multipartFile.getOriginalFilename()));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}

```

```

@Override
public void write(String fileName) throws IOException {
    File file = new File(fileName);
    if (!file.isAbsolute()) {
        file = new File(location, fileName);
    }
    try {
        fileItem.write(file);
    } catch (Exception e) {
        throw new IOException(e);
    }
}

public String getString(String encoding) throws UnsupportedOperationException {
    return fileItem.getString(encoding);
}

/*
 * Adapted from FileUploadBase.getFileName()
 */
@Override
public String getSubmittedFileName() {
    String fileName = null;
}

```

Multipart对于transferTo相对路径并不是项目的目录

```

@Override
public void transferTo(File dest) throws IOException, IllegalStateException {
    this.part.write(dest.getPath());
    if (dest.isAbsolute() && !dest.exists()) {
        // Servlet 3.0 Part.write is not guaranteed to support absolute file paths:
        // may translate the given path to a relative location within a temp dir
        // (e.g. on Jetty whereas Tomcat and Undertow detect absolute paths).
        // At least we offloaded the file from memory storage; it'll get deleted
        // from the temp dir eventually in any case. And for our user's purposes,
        // we can manually copy it to the requested location as a fallback.
        FileCopyUtils.copy(this.part.getInputStream(), Files.newOutputStream(dest.toPath()));
    }
}

```

- 如果想要使用相对路径，请手动获取InputStream，创建OutputStream，手动将数据写入到输出流。
 - JDK9中对InputStream提供了transferTo接口，可以直接以8KB的缓冲区大小将数据写入到输出流；

```

/**
 * 根据表单属性解析出Multipart，可以使用@RequestParam、也可以使用@RequestParam
 *
 * @param fileArray 多文件
 */
0 个用法 新 *
@PostMapping("/upload")
public void multiImageImport(@RequestParam("files") MultipartFile[] fileArray) {
    for (MultipartFile multipartFile : fileArray) {
        String filename = Objects.requireNonNull(multipartFile.getOriginalFilename());
        try {
            OutputStream os = new FileOutputStream(filename);
            InputStream is = multipartFile.getInputStream();

            // JDK9+
            // is.transferTo(os);

            final int BUFFER_SIZE = 8 * 1024;
            byte[] bytes = new byte[BUFFER_SIZE];
            int read;
            while ((read = is.read(bytes, off: 0, BUFFER_SIZE)) >= 0) {
                os.write(bytes, off: 0, read);
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}

```

附录1：Multipart文件在哪个Servlet中进行加载？

附录2：MIME解析（Spring-Web）

问这个问题之初，是为了找到spring内部有无描述所有MIME类型的类

- 最后发现文件上传的类型，是在form表选中文件后自动初始化，没有在后端进行解析MIME格式
 - <https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/Input/file>
 - https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- spring-web也提供了MIME解析类

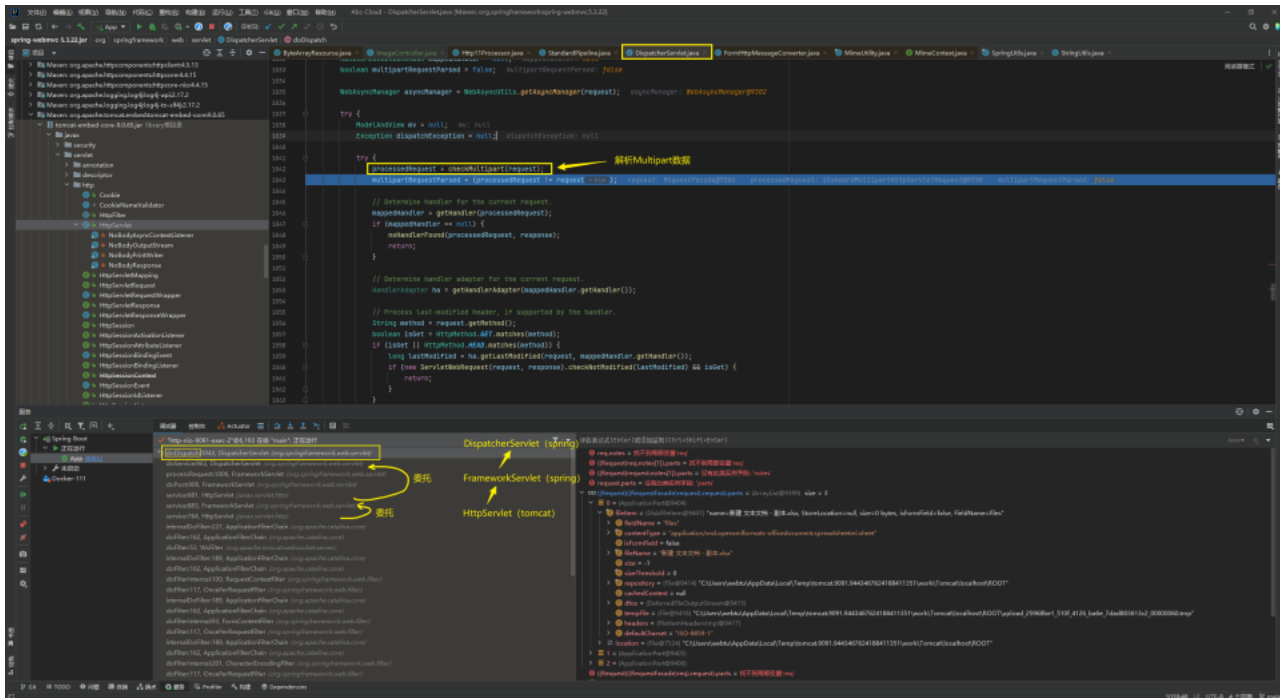
```
@SpringBootApplication
```

```
public class App {
```

0 个用法 trivis *

```
public static void main(String[] args) {
```

```
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.tzt").orElse(MediaType.TEXT_XML)); //
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.txt").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.png").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.gif").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.xlsx").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.pdf").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.ts").orElse(MediaType.TEXT_XML)); // X
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.flv").orElse(MediaType.TEXT_XML));
    System.out.println(MediaTypeFactory.getMediaType(filename: "aaa.mp4").orElse(MediaType.TEXT_XML));
    SpringApplication.run(App.class, args);
}
```



Convert the request into a multipart request, and make multipart resolver available.

If no multipart resolver is set, simply use the existing request.

形参: request – current HTTP request

返回值: the processed request (multipart wrapper if necessary)

请参阅: `MultipartResolver.resolveMultipart`

```
protected HttpServletRequest checkMultipart(HttpServletRequest request) throws MultipartException {
    if (this.multipartResolver != null && this.multipartResolver.isMultipart(request)) {
        if (WebUtils.getNativeRequest(request, MultipartHttpServletRequest.class) != null) {
            if (DispatcherType.REQUEST.equals(request.getDispatcherType())) {
                logger.trace("Request already resolved to MultipartHttpServletRequest, e.g. by MultipartFilter");
            }
        }
    }
    else if (hasMultipartException(request)) {
        logger.debug("Multipart resolution previously failed for current request - " +
            "skipping re-resolution for undisturbed error rendering");
    }
    else {
        try {
            return this.multipartResolver.resolveMultipart(request);
        }
        catch (MultipartException ex) {
            if (request.getAttribute(WebUtils.ERROR_EXCEPTION_ATTRIBUTE) != null) {
                logger.debug("message: \"Multipart resolution failed for error dispatch\", ex);
                // Keep processing error dispatch with regular request handle below
            }
            else {
                throw ex;
            }
        }
    }
    // If not returned before: return original request.
    return request;
}
```

multipart处理器

StandardMultipartHttpServletRequest.java

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

StandardMultipartHttpServletRequest.java

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

<

附录3：CORS后端跨域配置

```
import lombok.NonNull;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class CORSConfiguration {

    /**
     * 注意：跨域会导致页面获取不到response-headers!
     * @return
     */
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(@NonNull CorsRegistry registry) {
                registry.addMapping("/test/**")
                    .allowedOrigins("http://localhost:8880", "http://localhost:8881")
                    .allowedMethods("GET", "POST")
                    .allowCredentials(false).maxAge(3600);
            }
        };
    }
}
```