1. AJAX 와 JSON

가. Spring MVC JSON(ajax)

(1) 개발환경:

- Spring Framework
- Jackson
- jQuery

(2) pom.xml 설정

```
<!-- jackson -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.7.2</version>
</dependency>
```

(3) HttpMessageConver 등록

응답결과를 HTML이 아닌 JSON이나 XML로 변환하여 메시지 바디에 저장하려면 스프링에서 제공하는 변환기(convert)를 사용해야하므로 HttpMessageConvert를 구현한 클래스를 설정파일에 등록하여야 한다. 스프링 설정파일에 <annotation-driven>을 설정하면 HttpMessageConvert를 구현한 모든 변환기가 생성된다.

- servlet-context.xml 설정

```
<mvc:annotation-driven></mvc:annotation-driven>
```

(가) **<context:component-scan>** : 클래스를 스캔하고 빈 인스턴스 생성

- @Component
- @Repository, @Service, @Controller
- @Autowired

(나) **<mvc:annotation-driven>** : 스프링 MVC 컴포넌트들의 디폴트 설정을 적용

- @HandlerMapping 및 HandlerAdapter 등록
- @NumberFormat, @DateTimeFormat, @Valid
- Xml 및 JSON 읽고 쓰기(MessageConverter)

(다) **<context:annotation-config/>**

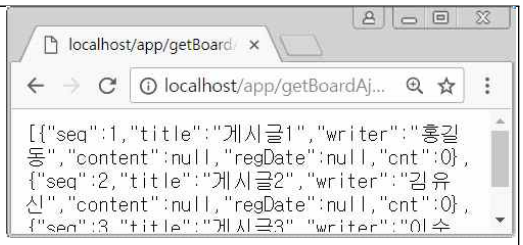
- @Required
- @Autowired
- @Resource, @PostConstruct, @PreDestory
- @Configuration

나. 서버에서 AJAX 요청 결과를 JSON string으로 응답

(1) controller

```
@RequestMapping("/ajaxTest.do")
@ResponseBody
public List<String> test(){
    List<String> list = new ArrayList<String>();
    list.add("홍길동");
    list.add("이순신");
    list.add("을지문덕");
    return list;
}
```

```
    @RequestMapping("/getBoardAjaxList.do")
    @ResponseBody
    public List<BoardVO> boardList(){
        List<BoardVO> list = new ArrayList<BoardVO>();
        list.add(new BoardVO(1,"게시글1","홍길동"));
        list.add(new BoardVO(2,"게시글2","김유신"));
        list.add(new BoardVO(3,"게시글3","이순신"));
        return list;
    }
```




```
@Controller
public class AjaxController {

    @Resource(name="deptService")
    DeptService deptService;

    @Resource(name="employeeService")
    EmployeeService employeeService;

    @RequestMapping("/ajaxTest.do")
    @ResponseBody
    public List<String> test(){
        List<String> list = new ArrayList<String>();
        list.add("홍길동");
        list.add("이순신");
        list.add("을지문덕");
        return list;
    }

    @RequestMapping("/ajaxDeptBeans.do")
    @ResponseBody
    public List<DeptBeans> ajaxDeptBeans (DeptBeans beans) {
        List<DeptBeans> list = new ArrayList<DeptBeans>();
        list.add(new DeptBeans("1000", "100", "총무", "10"));
        list.add(new DeptBeans("1100", "110", "기획", "20"));
        list.add(new DeptBeans("1200", "120", "인사", "30"));
        return list;
    }

    @RequestMapping("/ajaxDept.do")
    @ResponseBody
    public List<Map<String,Object>> ajaxDept (DeptBeans beans) {
        return deptService.selectAll(beans);
    }

    @RequestMapping("/ajaxDeptXml.do")
    @ResponseBody
    public DeptListVO ajaxDeptXml (DeptBeans beans) {
        List<DeptBeans> list = new ArrayList<DeptBeans>();
        list.add(new DeptBeans("1000", "100", "총무", "10"));
        list.add(new DeptBeans("1100", "110", "기획", "20"));
        list.add(new DeptBeans("1200", "120", "인사", "30"));

        DeptListVO deptListVO = new DeptListVO();
        deptListVO.setDeptList(list);
        return deptListVO;
    }

    //부서등록
    @RequestMapping(value="/employee/insertDepartments", method=RequestMethod.POST )
    public  @ResponseBody Map<String, ? extends Object>  insertDepartments(
            @RequestBody Departments bean,
            HttpServletResponse response) {
        employeeService.insertDepartments(bean);
        return Collections.singletonMap("result", bean.getDepartment_id());
    }

    //부서등록 유효성 체크
    @RequestMapping(value="/employee/availability", method=RequestMethod.GET)
    public @ResponseBody Map<String, ? extends Object> getAvailability(
                    @RequestParam String department_id) {
        String result ="true";
        Departments bean = employeeService.selectDepartment(department_id);
        if ( bean != null) result ="false";
        return Collections.singletonMap("result", result);
    }
}
```

다. 클라이언트에서 요청 파라미터를 쿼리문자열로 보내는 경우

Query에서 ajax() 함수의 기본 Content-Type은 "application/x-www-form-urlencoded"이므로 전송할 데이터는 query string 형식으로 작성해야 합니다. 쿼리스트링은 "key1=value&key2=value2&..."와 같은 포맷을 갖습니다.

1. form 요소 값들을 query string으로 만들기

```
$('#폼ID').serialize()
```

2. javascript array 객체를 query string으로 만들기

```
jQuery.param(array객체)
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Insert title here</title>
<script src="./resources/scripts/jquery-3.2.1.min.js"></script>
<script>
$(function(){

	$("#btnSave").click(function() {

		//쿼리문자열로 요청
		$.ajax({
			url : "./getParam.do",
			data : $("#frm").serialize(),   //쿼리문자열
			method : "post",
			type : "json",
			success : function(data) {
				$("#result").html("쿼리문자열로요청:"+data.writer)
			}
		});
	});
});
</script>
</head>
<body>
<div id="result">응답결과: </div>
<form id="frm">
	작성자<input type="text" name="writer"><br/>
	제목<input type="text" name="title"><br/>
	내용<textarea name="content"></textarea><br/>
	<input type="button" id="btnSave" value="저장">
</form>
</body>
</html>
```

라. 클라이언트에서 요청 파라미터를 Json String으로 보내는 경우

만약 요청 Content-Type을 "application/json" 으로 변경한 경우에는 쿼리스트링이 아니라 json 형식의 스트링으로 데이터를 전송해야 합니다.

(1) ajax 통신을 위해 jquery ajax method 설정

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script src="./resources/scripts/jquery-3.2.1.min.js"></script>
<script src="./resources/scripts/json.min.js"></script>
<script>

$(function(){

	$("#btnSave").click(function() {

		//쿼리문자열로 요청
		$.ajax({
			url : "./getParam.do",
			data : $("#frm").serialize(),   //쿼리문자열
			method : "post",
			type : "json",
			success : function(data) {
				console.log("쿼리문자열 파라미터");
				$("#result").html(data.writer)
			}
		});

		//json 타입으로 요청
		$.ajax({
			url : "./getParam.do",
			data : JSON.stringify( $("#frm").serializeObject() ) ,
			method : "post",
			type : "json",
			contentType : "application/json",
			success : function(data) {
				console.log(data);
				$("#result").html(data.writer)
			}
		});  */

		//함수 이용
		var p = $("#frm").serializeObject();
		$.postJSON("./getParam.do", p, function(data) {
			console.log("쿼리문자열 파라미터");
			$("#result").html(data.writer)
		});
	});
});
</script>
</head>
<body>
<div id="result">응답결과: </div>
<form id="frm">
	작성자<input type="text" name="writer"><br/>
	제목<input type="text" name="title"><br/>
	내용<textarea name="content"></textarea><br/>
	<input type="button" id="btnSave" value="저장">
</form>
</body>
</html>
```

```
<script type="text/javascript" src=<c:url value="/resources/jquery-2.2.3.js" /> ></script>
<script type="text/javascript" src="<c:url value="/resources/json.min.js" /> "></script>
<script>
$(document).ready(function() {
 $("#frm").submit(function() {
    var params =  $("#frm").serializeObject();
    $.postJSON("../employee/insertDepartments", params, function(data) {
        $("#result").html(data.result);
    });
    return false;
  });
});
</script>
```

(2) ajax 통신을 위해 jquery ajax method 설정

```
<script type="text/javascript" src=<c:url value="/resources/jquery-2.2.3.js" /> ></script>
<script type="text/javascript" src="<c:url value="/resources/json.min.js" /> "></script>
<script>
$(document).ready(function() {
 $("#frm").submit(function() {
    var params =  $("#frm").serialize();
    $.postJSON("../employee/insertDepartments", params, function(data) {
        $("#result").html(data.result);
    });
    return false;
  });
});
</script>
```

(3) controller

```
@RequestMapping(value="/employee/insertDepartments", method=RequestMethod.POST )
public  @ResponseBody Map<String, ? extends Object>  insertDepartments(
          @RequestBody Departments bean,
          HttpServletResponse response) {
          employeeDao.insertDepartments(bean);
          return Collections.singletonMap("result", bean.getDepartment_id());
}
```

@RequestBody는 클라이언트에서 jsonObject로 전송되는 파라미터를 자동으로 javaClass 형식으로 변환한다. (UserBean Class 는 클라이언트에서 받을 파라미터를 get, set 으로 작성하면, bean에 등록된 massageConverters에서 자동으로 javaClass 형식에 맞게 컨버팅이 이뤄진다.

@ResponseBody는 클라이언요청을 서버에서 처리 후 메소드가 리턴하는 오브젝트를 messageConverters를 통해 json 형태로 변환하여 리턴해주는 역할을 한다.

2. 파일업로드

파일 업로드와 같이 멀티파트 포멧의 요청정보를 처리하는 전략을 설정할 수 있다.

멀티파트 처리를 담당하는 다양한 구현으로 바꿀 수 있도록 설계되어 있지만, 현재는 아파치 Commons 의 FileUpload 라이브러리를 사용하는 CommonsMultipartResolver 한 가지만 지원된다.

가. 라이브러리 설치 (pom.xml)

```
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.2</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.2</version>
</dependency>
```

나. servlet-context.xml 에 multipartResolver 설정

멀티파트 리졸버 전략은 디폴트로 등록되는 것이 없다. 따라서 적용하려면 아래와 같이 빈을 등록해줘야 한다.

```
    <!-- MultipartResolver 설정-->
    <beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMu
ltipartResolver">
        <beans:property name="maxUploadSize" value="100000000" />
        <beans:property name="maxInMemorySize" value="100000000" />
    </beans:bean>
```

DispatcherServlet은 클라이언트로부터 멀티파트 요청을 받으면 멀티파트 리졸버에게 요청해서 HttpServletRequest 의 확장 타입인 MultipartHttpServletRequest 오브젝트로 전환한다.
MultipartHttpServletRequest 에는 멀티파트를 디코딩한 내용과 이를 참조하거나 조작할 수 있는 기능이 추가되어 있다.

다. 파일업로드(MultipartResolver)

(1) 컬럼추가

```
alter table board add ( uploadFileName varchar2(50) );  --첨부파일명
```

(2) VO에 필드 추가

```
    private String uploadFileName;    //첨부파일명
    private MultipartFile uploadFile;    //첨부파일

    public String getUploadFileName() {
        return uploadFileName;
    }
    public void setUploadFileName(String uploadFileName) {
        this.uploadFileName = uploadFileName;
    }
    public MultipartFile getUploadFile() {
        return uploadFile;
    }
    public void setUploadFile(MultipartFile uploadFile) {
        this.uploadFile = uploadFile;
    }
```

(3) view 페이지 수정

```
<form action="boardInsert.do" method="post"
      encType="multipart/form-data">
   작성자<input type="text" name="writer"><br/>
   제목<input type="text" name="title"><br/>
   내용<textarea name="content"></textarea><br/>
   첨부파일<input type="file" name="uploadFile"/><br/>
   <input type="submit" value="저장">
</form>
```

(4) 컨트롤러

```
    //등록
    @RequestMapping(value="/boardInsert.do",method=RequestMethod.POST)
    public String boardInsert(
            BoardVO vo,
            SessionStatus sessionStatus,
            HttpServletRequest request ) throws IllegalStateException, IOException{
        //첨부파일 업로드 처리
        MultipartFile uploadFile = vo.getUploadFile();
        String fileName = null;
        if(uploadFile !=null && !uploadFile.isEmpty() && uploadFile.getSize()>0) {
            fileName = uploadFile.getOriginalFilename();
            uploadFile.transferTo(new File("c:/upload/"+fileName));
        }
        //첨부파일명 VO에 지정
        vo.setUploadFileName(fileName);

        boardService.insertBoard(vo);
        sessionStatus.setComplete();
        return "redirect:/getBoardList.do";
    }
```

어노테이션 방식의 유연한 컨트롤러 메소드를 이용하면 처음부터 MultipartFile 타입의 파라미터로 전달받거나 변환기를 이용해서 아예 바이트 배열이나 파일 정보를 담은 오브젝트로 가져올 수도 있다.

(5) SQL 구문 수정

```
    <insert id="insertBoard" parameterType="board">

   <selectKey keyProperty="seq" resultType="int" order="BEFORE">
      SELECT NVL(MAX(SEQ), 0) + 1 FROM BOARD
    </selectKey>

        INSERT INTO BOARD(
        SEQ
        ,TITLE
        ,WRITER
        CONTENT
        <if test="uploadFileName != null and uploadFileName != ''">
        ,uploadFileName
        </if>
        )
        VALUES(
        #{seq}
        ,#{title}
        ,#{writer}
        ,#{content}
        <if test="uploadFileName != null and uploadFileName != ''">
        , #{uploadFileName}
        </if>
        )
    </insert>
```

라. MultipartHttpServletRequest 이용하여 파일 업로드

(1) 업로드 폼

- form 태그의 method 속성과 enctype 속성 지정

```
<form action="<c:url value="/main/addReply.do" /> " method="post"
enctype="multipart/form-data">
        <input type="text" name="ctnt" id="ctnt"/>
        <input type="text" name="writer" id="writer"/>
        <input type="file" name="image"/>
        <input type="file" name="atach"/>
        <input type="submit" value="저장"/>
</form>
```

(2) 파일 저장하기

아래와 같이 HttpServletRequest를 파라미터로 받은 컨트롤러에서는 전달받은 오브젝트를 MultipartHttpServletRequest 로 캐스팅한 후에 멀티파트 정보를 가진 MultipartFile 오브젝트를 가져와 사용할 수 있다.

```
public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) {

    MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest)request;
    MultipartFile multipartFile = multipartRequest.getFile("image");
    multipartFile.transferTo(new File("c:/upload", multipartFile.getOriginalFilename()));
    ....
}
```

마. BLOB 저장

(1) VO(Value Object) 작성

```
public class Reply {
        private String id;
        private String writer;
        private String ctnt;
        private String wdate;
        private byte[] imgData;    // blob 이미지
        private String atachFile;
        private String atachType;
        ...
}
```

blob 필드는 byte[]로 지정

(2) 컨트롤러

```
@RequestMapping(value="/main/addReply.do", method=RequestMethod.POST)
public String addReply(Reply reply, HttpServletRequest request) throws IOException{
                MultipartHttpServletRequest multipartRequest =
(MultipartHttpServletRequest)request;
        //이미지파일
        MultipartFile multipartFile = multipartRequest.getFile("image");
        reply.setImgData(multipartFile.getBytes());
        //첨부파일
        multipartFile = multipartRequest.getFile("atach");
        if(! multipartFile.isEmpty()) {
                multipartFile.transferTo(new File("c:/upload",multipartFile.getOriginalFilename()));
                reply.setAtchFile(multipartFile.getOriginalFilename());
                reply.setAtchType(multipartFile.getContentType());
        }
        mainService.insertReply(reply);
        return "main/main";
}
```

multipartFile.getBytes()로 읽어서 VO에 담아서 저장한다.

바. BLOB 이미지 읽기

(1) view

```
<img src="./getByteImage.do?id=${reply.id}"/>
<c:if test="${not empty reply.atachFile}">
        <a href="./FileDown.do?id=${reply.id}">${reply.atachFile}</a>
</c:if>
```

(2) 컨트롤러

(가) ResponseEntity 이용

```
@RequestMapping(value="/main/getByteImage2.do")
public ResponseEntity<byte[]> getByteImage2(Reply reply, HttpServletResponse response) throws
SerialException, Exception {
        Reply vo = mainService.selectReplyOne(reply);
        final HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.IMAGE_PNG);
        return new ResponseEntity<byte[]>(vo.getImgData(), headers, HttpStatus.OK);
}
```

(나) outputStream 이용

```
@RequestMapping(value="/main/getByteImage.do")
public void getByteImage(Reply reply, HttpServletResponse response) throws SerialException,
Exception {
        Reply vo = mainService.selectReplyOne(reply);
        SerialBlob blob = new SerialBlob(vo.getImgData());
        ServletOutputStream outputStream = response.getOutputStream();
        IOUtils.copy(blob.getBinaryStream(), outputStream);
        outputStream.flush();
        outputStream.close();
}
```

```
        //byte[]
encoded=org.apache.commons.codec.binary.Base64.encodeBase64(vo.getProfileImg());


  /*    SerialBlob blob = new SerialBlob(vo.getProfileImg());
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        byte[] buf = new byte[1024];
        InputStream in = blob.getBinaryStream();
        int n = 0;
        while ((n=in.read(buf))>=0)
        {
            baos.write(buf, 0, n);
        }
        in.close();
*/
/*        byte[] bytes = baos.toByteArray();
        System.out.println("bytes" +bytes);
*/
/*        byte[] encodeBase64 = Base64.encodeBase64(buf);
        String base64Encoded = new String(encodeBase64, "UTF-8");
*/
        //vo.setImgStr(base64Encoded);
```

(다) img 태그

```
<img src="data:image/jpeg;bytes,${profileVO.profileImg}"/>
```

참고사이트

http://stackoverflow.com/questions/27145424/converting-blob-image-to-array-to-display-in-the-web-page-in-spring-mvc

사. 파일다운로드

```
@Controller
public class EgovFileDownloadController {

    /**
     * 브라우저 구분 얻기.
     *
     * @param request
```

```
     * @return
     */
    private String getBrowser(HttpServletRequest request) {
        String header = request.getHeader("User-Agent");
        if (header.indexOf("MSIE") > -1) {
            return "MSIE";
        } else if (header.indexOf("Trident") > -1) { // IE11 문자열 깨짐 방지
            return "Trident";
        } else if (header.indexOf("Chrome") > -1) {
            return "Chrome";
        } else if (header.indexOf("Opera") > -1) {
            return "Opera";
        }
        return "Firefox";
    }

    /**
     * Disposition 지정하기.
     *
     * @param filename
     * @param request
     * @param response
     * @throws Exception
     */
    private void setDisposition(String filename, HttpServletRequest request, HttpServletResponse response)
            throws Exception {
        String browser = getBrowser(request);

        String dispositionPrefix = "attachment; filename=";
        String encodedFilename = null;

        if (browser.equals("MSIE")) {
            encodedFilename = URLEncoder.encode(filename, "UTF-8").replaceAll("₩₩+", "%20");
        } else if (browser.equals("Trident")) { // IE11 문자열 깨짐 방지
            encodedFilename = URLEncoder.encode(filename, "UTF-8").replaceAll("₩₩+", "%20");
        } else if (browser.equals("Firefox")) {
            encodedFilename = "₩"" + new String(filename.getBytes("UTF-8"), "8859_1") + "₩"";
        } else if (browser.equals("Opera")) {
            encodedFilename = "₩"" + new String(filename.getBytes("UTF-8"), "8859_1") + "₩"";
        } else if (browser.equals("Chrome")) {
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < filename.length(); i++) {
                char c = filename.charAt(i);
                if (c > '~') {
                    sb.append(URLEncoder.encode("" + c, "UTF-8"));
                } else {
                    sb.append(c);
                }
            }
            encodedFilename = sb.toString();
        } else {
            throw new IOException("Not supported browser");
        }

        response.setHeader("Content-Disposition", dispositionPrefix + encodedFilename);

        if ("Opera".equals(browser)) {
            response.setContentType("application/octet-stream;charset=UTF-8");
        }
    }

    /**
     * 첨부파일로 등록된 파일에 대하여 다운로드를 제공한다.
     *
     * @param commandMap
     * @param response
     * @throws Exception
     */
    @RequestMapping(value = "/FileDown.do")
    public void cvplFileDownload(@RequestParam Map<String, Object> commandMap, HttpServletRequest
request,
            HttpServletResponse response) throws Exception {

        String atchFileId = (String) commandMap.get("atchFileId");

        File uFile = new File("c:/upload", atchFileId);
        long fSize = uFile.length();

        if (fSize > 0) {
            String mimetype = "application/x-msdownload";

            response.setContentType(mimetype);
```

```
            // response.setHeader("Content-Disposition", "attachment;
            // filename=₩"" + URLEncoder.encode(fvo.getOrignlFileNm(), "utf-8") + "₩"");
            setDisposition(atchFileId, request, response);

            BufferedInputStream in = null;
            BufferedOutputStream out = null;

            try {
                in = new BufferedInputStream(new FileInputStream(uFile));
                out = new BufferedOutputStream(response.getOutputStream());

                FileCopyUtils.copy(in, out);
                out.flush();
            } catch (IOException ex) {

            } finally {
                in.close();
                response.getOutputStream().flush();
                response.getOutputStream().close();
            }

        } else {
            response.setContentType("application/x-msdownload");

            PrintWriter printwriter = response.getWriter();

            printwriter.println("<html>");
            printwriter.println("<h2>Could not get file name:<br>" + atchFileId + "</h2>");
            printwriter.println("<center><h3><a href='javascript: history.go(-1)'>Back</a></h3></center>");
            printwriter.println("&copy; webAccess");
            printwriter.println("</html>");

            printwriter.flush();
            printwriter.close();
        }
    }
}
```

3. AJAX 파일업로드

가. js 파일 다운로드

url: http://malsup.com/jquery/form/

파일명: jquery.form.min.js

나. 업로드 폼

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>ajax fileupload</title>
<script src="../scripts/jquery-3.1.1.min.js"></script>
<script src="../scripts/jquery.form.min.js"></script>
<script>
$(document.body).ready(function(){
		$("#insertBtn").on("click", function(){
			$("#frm").ajaxForm({
				dataType:"json",
				data:{writer:"admin"},
				url:'../ajaxBoardInsert.do',
			success: function(result, textStatus){
			  if(result.code = 'success') {
			    alert("등록되었습니다.");
			  }
			},
			error: function(){
				alert("파일업로드 중 오류가 발생하였습니다.");
				return;
			}
			  }).submit();
		});
	});
</script>
</head>
<body>
 <form id="frm" name="frm" method="post">
    작성자<input type="text" name="writer"><br/>
		제목<input type="text" name="title"><br/>
		내용<textarea name="content"></textarea><br/>
		첨부파일<input type="file" name="uploadFile"/>
    <input type="file" id="uploadFile" name="uploadFile"><br />
    <a href="javascript:;" id="insertBtn">저장</a>
 </form>
</body>
</html>
```

다. 컨트롤러 작성

```
@Controller
public class AjaxController {

	String UPLOAD_DIR = "d:/upload";

	@Autowired BoardService boardService;

	//ajax 등록
	@RequestMapping(value="/ajaxBoardInsert.do" , method=RequestMethod.POST)
	@ResponseBody
	public Map insertBoard(BoardVO vo)	//command 객체
			throws IllegalStateException, IOException {
			MultipartFile file = vo.getUploadFile();
			File saveFile = new File("d:/upload/",file.getOriginalFilename());
			file.transferTo(saveFile);	//서버에 파일 저장
			vo.setAttatchFilename(file.getOriginalFilename()); //파일명 저장
			boardService.insertBoard(vo);
			System.out.println(vo);
			Map<String,String> map = new HashMap<String,String>();
			map.put("code", "success");
			return map;
	}

}
```

4. 구글차트

가. 구글차트 사용하기

(1) jsapi

```
<script src="//www.google.com/jsapi"></script>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>Insert title here</title>
<script src="//www.google.com/jsapi"></script>
<script>
var data = [
  ['원소', '밀도'],
  ['구리', 8.94],
  ['은', 10.49],
  ['금', 19.30],
  ['백금', 21.45],
];
var options = {
  title: '귀금속 밀도 (단위: g/cm³)',
  width: 400, height: 500
};
google.load('visualization', '1.0', {'packages':['corechart']});
google.setOnLoadCallback(function() {
  var chart = new google.visualization.ColumnChart(document.querySelector('#chart_div'));
  chart.draw(google.visualization.arrayToDataTable(data), options);
});
</script>
</head>
<body>
<div id="chart_div"></div>
</body>
</html>
```

(2) loader

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
```

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">

      // Load the Visualization API and the corechart package.
      google.charts.load('current', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.charts.setOnLoadCallback(drawChart);

      // Callback that creates and populates a data table,
      // instantiates the pie chart, passes in the data and draws it.
      function drawChart() {

        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'Topping');
        data.addColumn('number', 'Slices');
        data.addRows([
```

```
            ['Mushrooms', 3],
            ['Onions', 1],
            ['Olives', 1],
            ['Zucchini', 1],
            ['Pepperoni', 2]
          ]);

          // Set chart options
          var options = {'title':'How Much Pizza I Ate Last Night',
                         'width':400,
                         'height':300};

          // Instantiate and draw our chart, passing in some options.
          var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
          chart.draw(data, options);
        }
    </script>
  </head>

  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div"></div>
  </body>
</html>
```

나. spring에서 사용하기

(1) pom.xml

```
    <dependency>
        <groupId>com.googlecode.charts4j</groupId>
        <artifactId>charts4j</artifactId>
        <version>1.3</version>
    </dependency>
```

(2) 컨트롤러

```
    //차트
    @RequestMapping(value="/chart.do")
    public String chart(ModelMap model)
            throws Exception {
        Slice s1 = Slice.newSlice(15, Color.newColor("CACACA"),"mac");
        Slice s2 = Slice.newSlice(50, Color.newColor("DF7417"),"window");
        Slice s3 = Slice.newSlice(25, Color.newColor("951800"),"linux");
        Slice s4 = Slice.newSlice(10, Color.newColor("01A1DB"),"others");
        PieChart pieChart = GCharts.newPieChart(s1,s2,s3,s4);
        pieChart.setTitle("google chart", Color.BLACK,15);
        pieChart.setSize(720, 360);
        pieChart.setThreeD(true);
        model.addAttribute("pieUrl",pieChart.toURLString());
        return "/chart/googlechart3";
    }
```

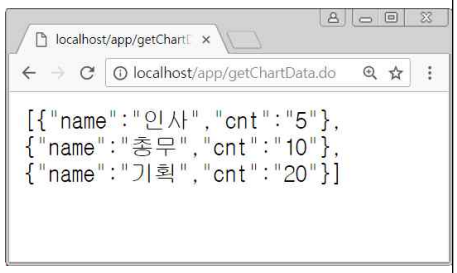
(3) 뷰

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>
<html>
  <head>
        <title>google chart</title>
  </head>

  <body>
  차트
    <img alt="google pie chart" src="${pieUrl}"/>
  </body>
</html>
```

다. AJAX 연동

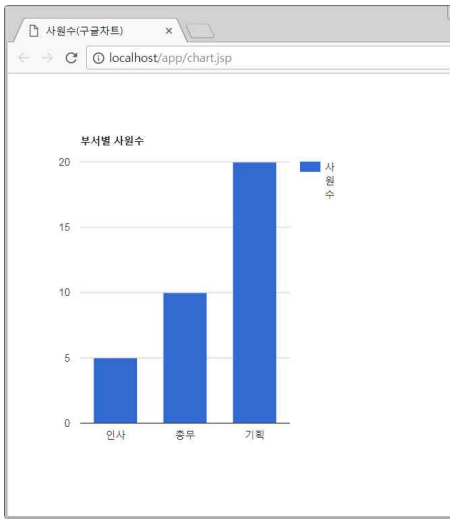
(1) 컨트롤러(ajax 요청 처리)

```
@RequestMapping("/getChartData.do")
@ResponseBody
public List<Map<String, String>> getChartData(){
    List<Map<String, String>> list =
            new ArrayList<Map<String, String>>();
    Map<String, String> map = new HashMap<String, String>();
    map.put("name", "인사");
    map.put("cnt", "5");
    list.add(map);

    map = new HashMap<String, String>();
    map.put("name", "총무");
    map.put("cnt", "10");
    list.add(map);

    map = new HashMap<String, String>();
    map.put("name", "기획");
    map.put("cnt", "20");
    list.add(map);
    return list;
}
```



(2) 뷰페이지(ajax 요청)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>사원수(구글차트)</title>
<script src="//www.google.com/jsapi"></script>
<script src="./resources/scripts/jquery-3.2.1.min.js"></script>
<script src="./resources/scripts/json.min.js"></script>
<script>
var options = {
  title: '부서별 사원수',
  width: 400,
  height: 500
};
google.load('visualization', '1.0', {'packages':['corechart']});
google.setOnLoadCallback(function() {
    //차트에 넣을 data를 ajax 요청해서 가져옴
    $.ajax({
        url : "./getChartData.do",
        method : "post",
        type : "json",
        success : function(data) {
            //ajax결과를 chart에 맞는 data 형태로 가공
            var chartData = [];
            chartData.push(['사원명','사원수'])
            for(i=0; i<data.length; i++) {
                var subarr = [data[i].name, parseInt(data[i].cnt) ];
                chartData.push(subarr);
            }
            //챠트 그리기
            var chart = new
        google.visualization.ColumnChart(document.querySelector('#chart_div'));
            chart.draw(google.visualization.arrayToDataTable(chartData), options);
        }
    });

});
</script>
</head>
<body>
<div id="chart_div"></div>
</body>
</html>
```



라. 구글차트 API

URL : https://developers.google.com/chart/