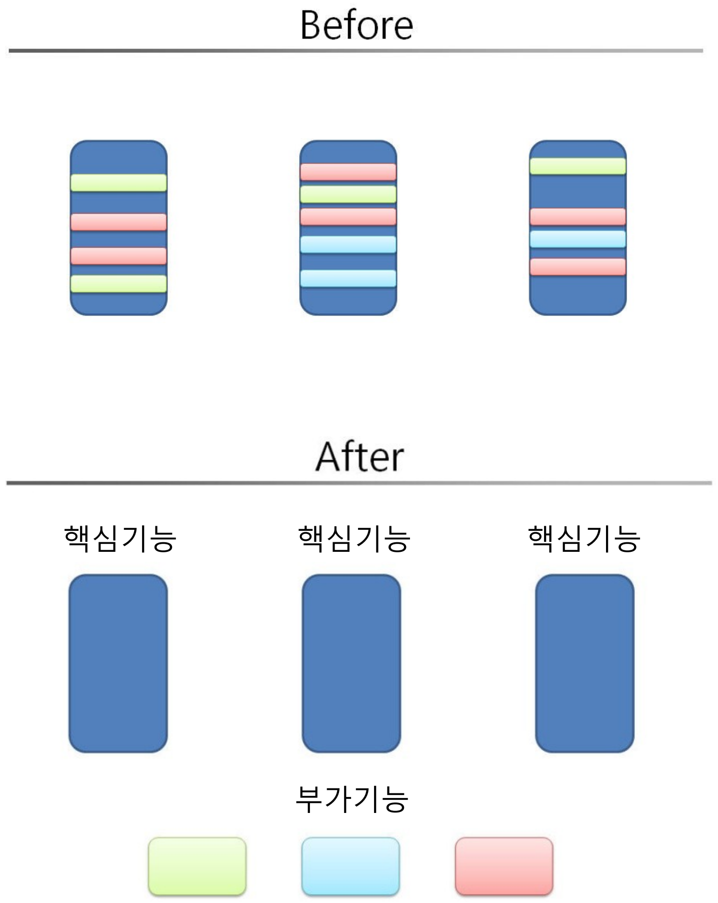
■ AOP

- Aspect Oriented Programming
- 공통적으로 적용될 모듈(클래스/메소드)을 만든 후 적용하고자 하는 부분의 코드 밖에서 삽입하는 방법
- 사용 분야
  - 메소드의 성능 테스트
  - 트랜잭션 처리
  - 예외 반환
  - 로깅, 인증, 권한 처리



■ AOP

용어	설명
Target	부가기능을 부여할 대상 (핵심기능을 담은 클래스 등)
Advice	**부가기능을 담은 모듈**
Joinpoint	Advice가 적용될 수 있는 지점
Pointcut	**Joinpoint 중 실제 Advice가 적용되는 지점**
Proxy	Advice를 Target에 적용할때(Weaving) 생기는 객체 메소드(Pointcut) 호출을 대신 받아서 Target에 위임
Advisor	Pointcut과 Advice를 하나씩 가지고 있는 오브젝트
Weaving	Advice를 Target에 적용하는 과정 부가기능을 핵심기능에 삽입
Aspect	**Advisor의 집합** **여러 객체에 공통으로 적용되는 공통 관심 사항**

■ AOP

용어	설명
before	메소드 호출 전
after-returning	메소드 호출 후 (정상종료)
after-throwing	메소드 실행 중 오류 발생 시
after	메소드 호출 후 (실행결과와 관계없이 항상 실행)
around	메소드 호출 전 / 후

■ AOP

- **execution() : 적용할 메소드를 세부적으로 명시**

  execution(public void set*(..))

  ➔ 리턴 타입 : void , 메소드명 : set으로 시작, 파라미터 : 0개 이상

  execution(* com.spring.aop.*.*())

  ➔ 리턴 타입 : All, 메소드명 : All, 파라미터 : 없음

  execution(* com.spring.aop..First.process(..))

  ➔ 리턴 타입 : All, 메소드명 : process, 파라미터 : 0개 이상

  execution(String com.spring.aop.exam.First.process())

  ➔ 리턴 타입 : String, 메소드명 : process, 파라미터 : 없음

  execution(* get*(*))

  ➔ 리턴 타입 : All, 메소드명 : get으로 시작, 파라미터 : 1개

  execution(* get*(*, *))

  ➔ 리턴 타입 : All, 메소드명 : get으로 시작, 파라미터 : 2개

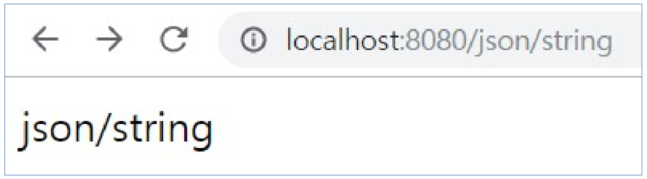
■ AOP

- aspect/ControllerAspect.java

```java
package com.ggoreb.basic.aspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Aspect
@Component
public class ControllerAspect {
  @Before(value = "execution (* com.ggoreb.basic.controller.*.*(..))")
  public void onBeforeHandler(JoinPoint joinPoint) {
    log.debug("@Before run");
  }
  @After(value = "execution (* com.ggoreb.basic.controller.*.*(..))")
  public void onAfterHandler(JoinPoint joinPoint) {
    log.debug("@After run");
  }
  @AfterReturning(value = "execution (* com.ggoreb.basic.controller.*.*(..))"
          , returning = "data")
  public void onAfterReturningHandler(JoinPoint joinPoint, Object data) {
    if(data != null) {
      log.debug(data.toString());
    }
    log.debug("@AfterReturning run");
  }
}
```

■ AOP

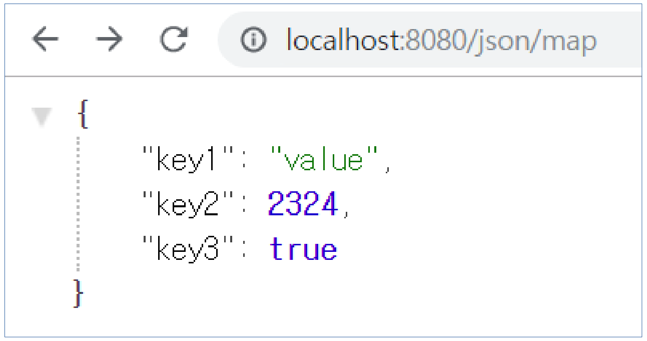
● http://localhost:8080/json/string


localhost:8080/json/string

json/string

```
c.ggoreb.basic.aspect.ControllerAspect    : @Before run
c.ggoreb.basic.aspect.ControllerAspect    : @After run
c.ggoreb.basic.aspect.ControllerAspect    : json/string
c.ggoreb.basic.aspect.ControllerAspect    : @AfterReturning run
```

● http://localhost:8080/json/map

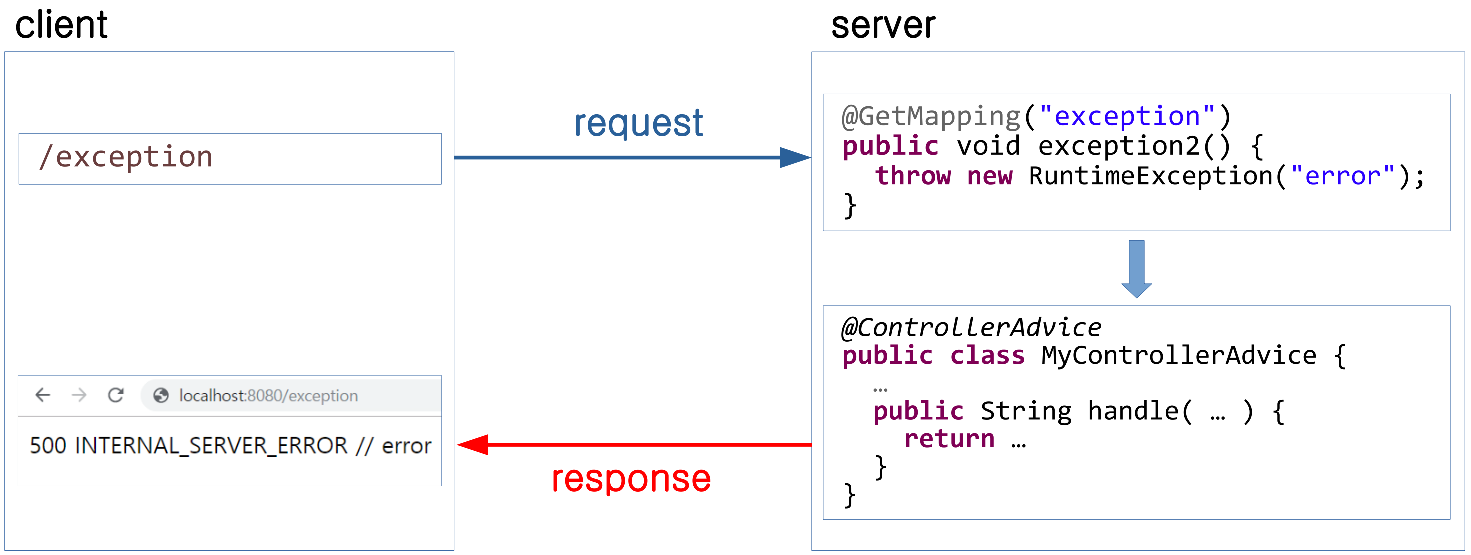

localhost:8080/json/map

```
{
    "key1": "value",
    "key2": 2324,
    "key3": true
}
```

```
c.ggoreb.basic.aspect.ControllerAspect    : @Before run
c.ggoreb.basic.aspect.ControllerAspect    : @After run
c.ggoreb.basic.aspect.ControllerAspect    : {key1=value, key2=2324, key3=true}
c.ggoreb.basic.aspect.ControllerAspect    : @AfterReturning run
```

■ ControllerAdvice

- Controller에서 발생되는 오류를 감지하고 처리해주는 기능
- 사용 이유
  - 예외처리를 한 곳에 묶어서 편하게 관리
  - 처리가 제대로 되지 못한 부분에 예외가 발생되는 경우 브라우저에 Exception Message가 노출되어 버리는데 모든 예측하지 못한 예외도 한꺼번에 처리 가능



```
500 INTERNAL_SERVER_ERROR // error
```

```
@GetMapping("exception")
public void exception2() {
  throw new RuntimeException("error");
}
```

```
@ControllerAdvice
public class MyControllerAdvice {
  …
  public String handle( … ) {
    return …
  }
}
```

■ ControllerAdvice

● 기본 구조

```
@ControllerAdvice
public class MyControllerAdvice {
  @ExceptionHandler
  [@ResponseStatus]
  [@ResponseBody]
  public String handle(RuntimeException e, WebRequest request) {
    return [view];
  }
}
```

● @ControllerAdvice의 옵션으로 특정 패키지, 특정 클래스만 지정 가능

– 기본값 : 프로젝트 기본 패키지 내의 모든 컨트롤러

● @ExceptionHandler의 옵션으로 특정 Exception에 대해서만 동작 가능

– 기본값 : Exception (모든 예외)

● @ResponseStatus의 옵션으로 응답 코드 지정 가능

– 기본값 : HttpStatus.INTERNAL_SERVER_ERROR (500)

● 메소드의 리턴 타입은 컨트롤러에서 사용하는 것과 동일

■ ControllerAdvice

- exception/MyException.java

```
package com.ggoreb.basic.exception;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class MyException extends RuntimeException {
	private String message;
}
```

■ ControllerAdvice

- controller/ExceptionController.java

```java
package com.ggoreb.basic.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ggoreb.basic.exception.MyException;

@RestController
public class ExceptionController {
	@GetMapping("exception1")
	public String exception1() throws Exception {
		throw new Exception("exception!");
//		return "exception1";
	}
	@GetMapping("exception2")
	public String exception2() {
		throw new MyException("runtime exception!");
//		return "exception2";
	}
}
```

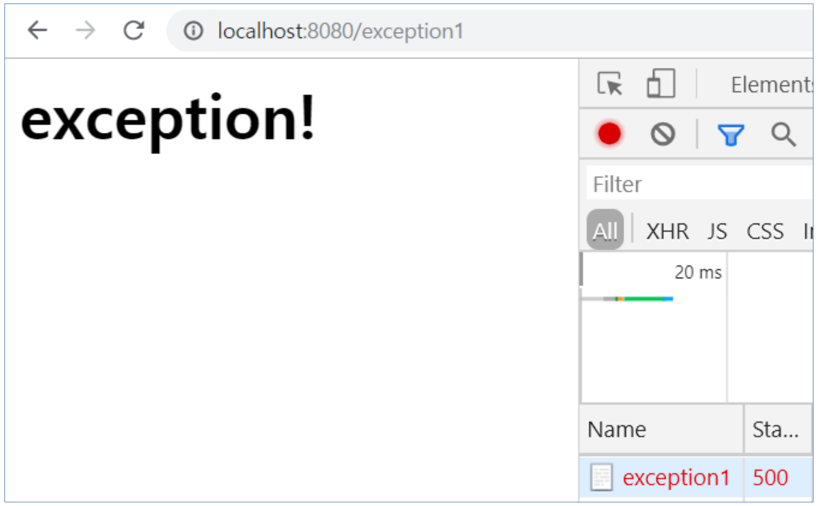
■ ControllerAdvice

- aspect/MyControllerAdvice.java

```java
package com.ggoreb.basic.aspect;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.context.request.WebRequest;

import com.ggoreb.basic.controller.ExceptionController;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@ControllerAdvice(assignableTypes = {ExceptionController.class})
public class MyControllerAdvice {
    @ExceptionHandler
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    @ResponseBody
    public String handle(Exception e, WebRequest request) {
        String message = e.getMessage();
        log.debug(message);
        return "<h1>" + message + "</h1>";
    }
}
```

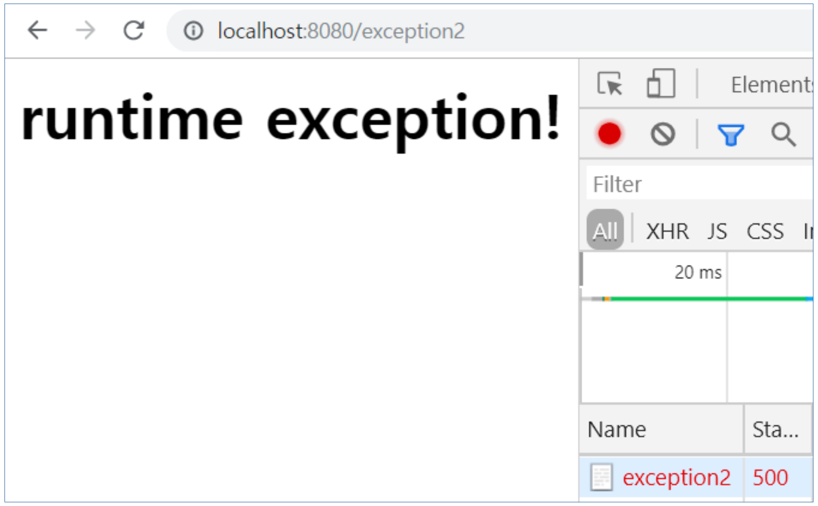
■ ControllerAdvice

- http://localhost:8080/exception1


localhost:8080/exception1

exception!

- http://localhost:8080/exception2

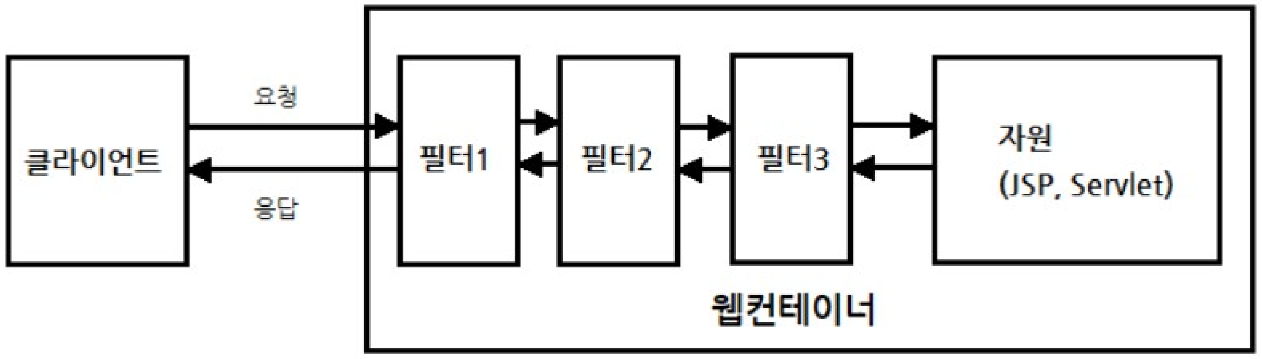

localhost:8080/exception2

runtime exception!

■ Filter

● HTTP 요청과 응답을 변경 할 수 있는 클래스

● 사용 분야
- XSS (Cross Site Scripting) 방지
- Logging
- Encoding
- IP 검사 등



■ Filter

- filter/IPCheckFilter.java

```java
package com.ggoreb.basic.filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;

import lombok.extern.slf4j.Slf4j;

@Slf4j
public class IPCheckFilter implements Filter {
	@Override
	public void doFilter(
			ServletRequest request, ServletResponse response, FilterChain chain)
			 throws IOException, ServletException {
		log.debug("filter begin");

		HttpServletRequest req = (HttpServletRequest) request;
		String ip = request.getRemoteAddr();
		log.debug("ip : " + ip);
		chain.doFilter(req, response);

		log.debug("filter end");
	}
}
```

■ Filter

● config/FilterConfig.java

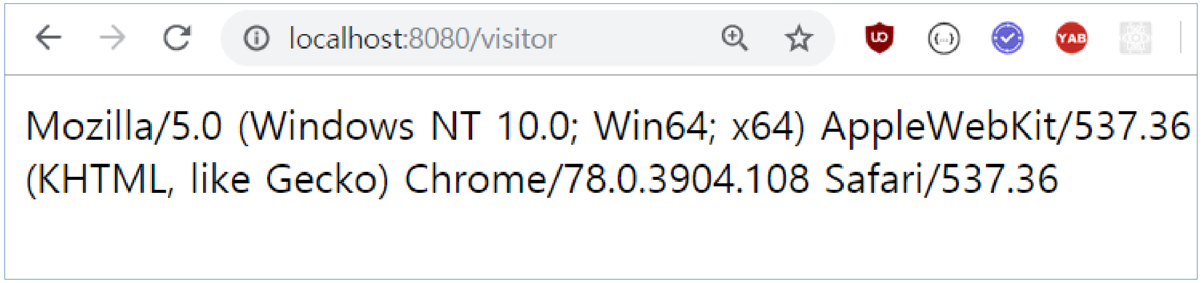
```
package com.ggoreb.basic.config;

import javax.servlet.Filter;

import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.ggoreb.basic.filter.IPCheckFilter;

@Configuration
public class FilterConfig {
	@Bean
	public FilterRegistrationBean<Filter> getFilterRegistrationBean() {
		FilterRegistrationBean<Filter> bean =
				new FilterRegistrationBean<>(new IPCheckFilter());
		bean.addUrlPatterns("/visitor");
		return bean;
	}
}
```

- 모든 주소 지정 시 : /*

■ Filter

● controller/VisitorController.java

```java
package com.ggoreb.basic.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class VisitorController {
	@GetMapping("/visitor")
	public String visitor(@RequestHeader("user-agent") String userAgent) {
		return userAgent;
	}
}
```

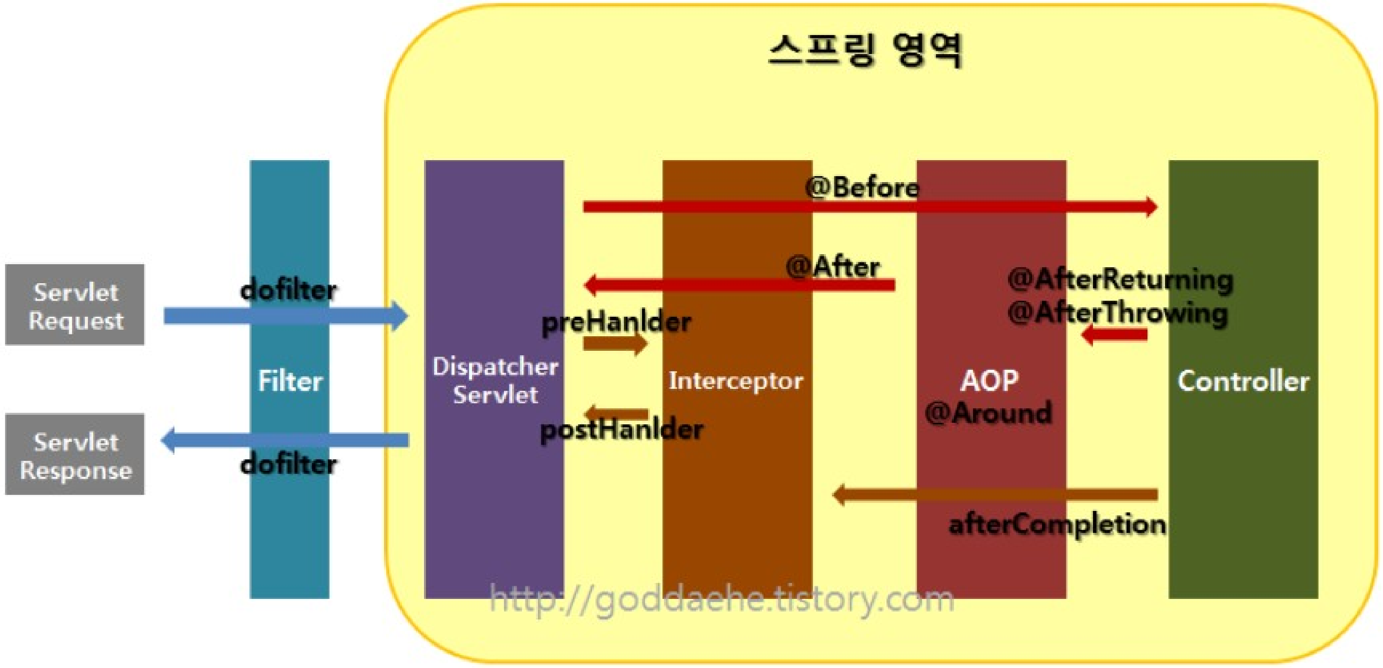


Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36

```
com.ggoreb.basic.filter.IPCheckFilter    : filter begin
com.ggoreb.basic.filter.IPCheckFilter    : ip : 0:0:0:0:0:0:0:1
c.ggoreb.basic.aspect.ControllerAspect   : @Before run
c.ggoreb.basic.aspect.ControllerAspect   : @After run
c.ggoreb.basic.aspect.ControllerAspect   : Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
c.ggoreb.basic.aspect.ControllerAspect   : @AfterReturning run
com.ggoreb.basic.filter.IPCheckFilter    : filter end
```

■ Interceptor

- Controller에 들어오는 요청 및 응답을 가로채는 역할
- Filter와 유사하지만 동작하는 시기가 다름
- 주요 메소드
  - preHandler() : Controller의 메소드가 실행되기 전 (요청)
  - postHandler() : Controller의 메소드가 실행된 후 (응답)
  - afterCompletion() – View가 Rendering 된 이후



■ Interceptor

- interceptor/SignInCheckInterceptor.java

```java
package com.ggoreb.basic.interceptor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Component;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

import com.ggoreb.basic.model.User;

import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j
public class SignInCheckInterceptor extends HandlerInterceptorAdapter {
  @Override
  public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
        Object handler) throws Exception {
    log.debug("preHandle");
    HttpSession session = request.getSession();
    User user = (User) session.getAttribute("user");
    if(user == null) {
      response.sendRedirect("/login");
    }
    return super.preHandle(request, response, handler);
  }
```

■ Interceptor

● interceptor/SignInCheckInterceptor.java

```
   @Override
   public void postHandle(HttpServletRequest request, HttpServletResponse response,
         Object handler, ModelAndView modelAndView) throws Exception {
      log.debug("postHandle");
      super.postHandle(request, response, handler, modelAndView);
   }

   @Override
   public void afterCompletion(HttpServletRequest request, HttpServletResponse response,
         Object handler, Exception ex) throws Exception {
      log.debug("afterCompletion");
      super.afterCompletion(request, response, handler, ex);
   }
}
```

■ Interceptor

● config/SignInCheckInterceptor.java

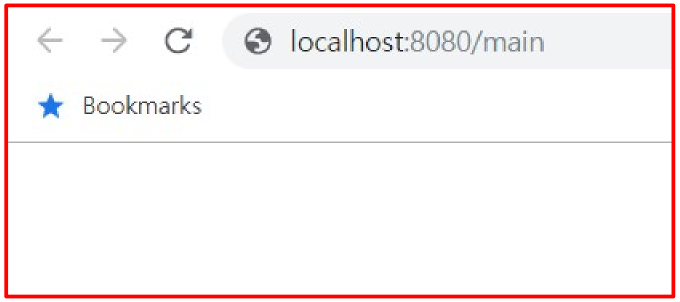
```
package com.ggoreb.basic.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import com.ggoreb.basic.interceptor.SignInCheckInterceptor;

@Configuration
public class InterceptorConfig implements WebMvcConfigurer {
  @Autowired
  private SignInCheckInterceptor signInCheckInterceptor;

  @Override
  public void addInterceptors(InterceptorRegistry registry) {
    registry.addInterceptor(signInCheckInterceptor).addPathPatterns("/main");

    WebMvcConfigurer.super.addInterceptors(registry);
  }
}
```

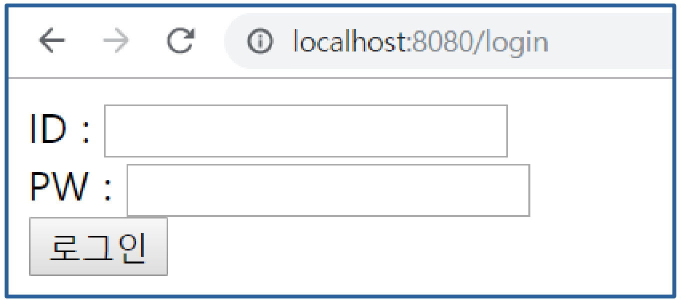
- 모든 주소 지정 시 : /**

- 특정 주소 제외 : excludePathPatterns("url")

■ Interceptor

- http://localhost:8080/main



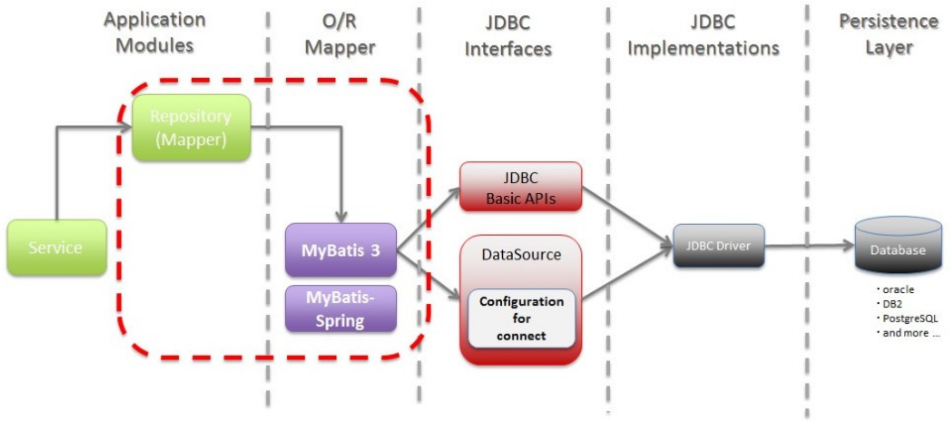


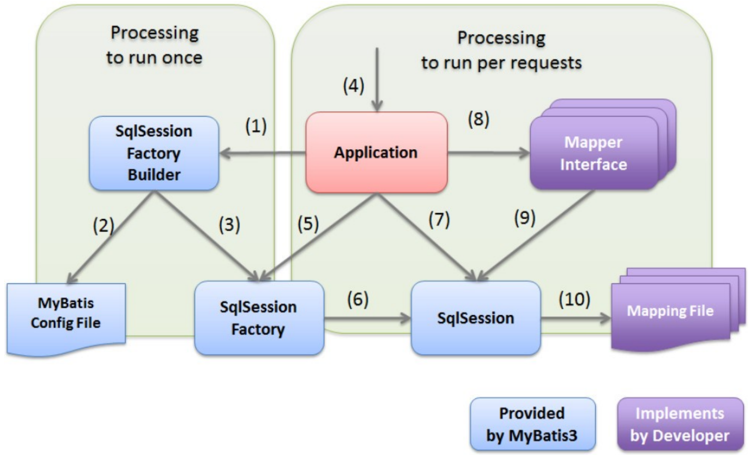
```
com.ggoreb.basic.filter.IPCheckFilter      : filter begin
com.ggoreb.basic.filter.IPCheckFilter      : ip : 0:0:0:0:0:0:0:1
c.g.b.i.SignInCheckInterceptor             : preHandle
c.ggoreb.basic.aspect.ControllerAspect     : @Before run
c.ggoreb.basic.aspect.ControllerAspect     : @After run
c.ggoreb.basic.aspect.ControllerAspect     : main
c.ggoreb.basic.aspect.ControllerAspect     : @AfterReturning run
c.g.b.i.SignInCheckInterceptor             : postHandle
c.g.b.i.SignInCheckInterceptor             : afterCompletion
com.ggoreb.basic.filter.IPCheckFilter      : filter end
```

```
com.ggoreb.basic.filter.IPCheckFilter      : filter begin
com.ggoreb.basic.filter.IPCheckFilter      : ip : 0:0:0:0:0:0:0:1
c.ggoreb.basic.aspect.ControllerAspect     : @Before run
c.ggoreb.basic.aspect.ControllerAspect     : @After run
c.ggoreb.basic.aspect.ControllerAspect     : login
c.ggoreb.basic.aspect.ControllerAspect     : @AfterReturning run
com.ggoreb.basic.filter.IPCheckFilter      : filter end
```

■ MyBatis

- ● 자바에서 관계형 데이터베이스를 좀 더 쉽게 개발할 수 있도록 하는 프레임워크
- ● SQL 문장들을 별도의 파일로 구성 (SQL 분리)
- ● 특징
  - 동적 SQL 지원
  - 프로그램 코드와 SQL의 분리로 코드의 간결성 및 유지보수성 향상





■ MyBatis

● SQL 코드

– mapper.xml

```
<select id="select" parameterType="MemberDto"  resultType="MemberDto">
    SELECT M_ID, M_PW, M_NAME, CRE_DATE
      FROM JDBC_MEMBER
     WHERE M_ID = #{mId}
</select>
```

● 프로그램 코드

– Dao.java

```
public interface MemberDao {
    public MemberDto select(MemberDto memberDto);
}
```

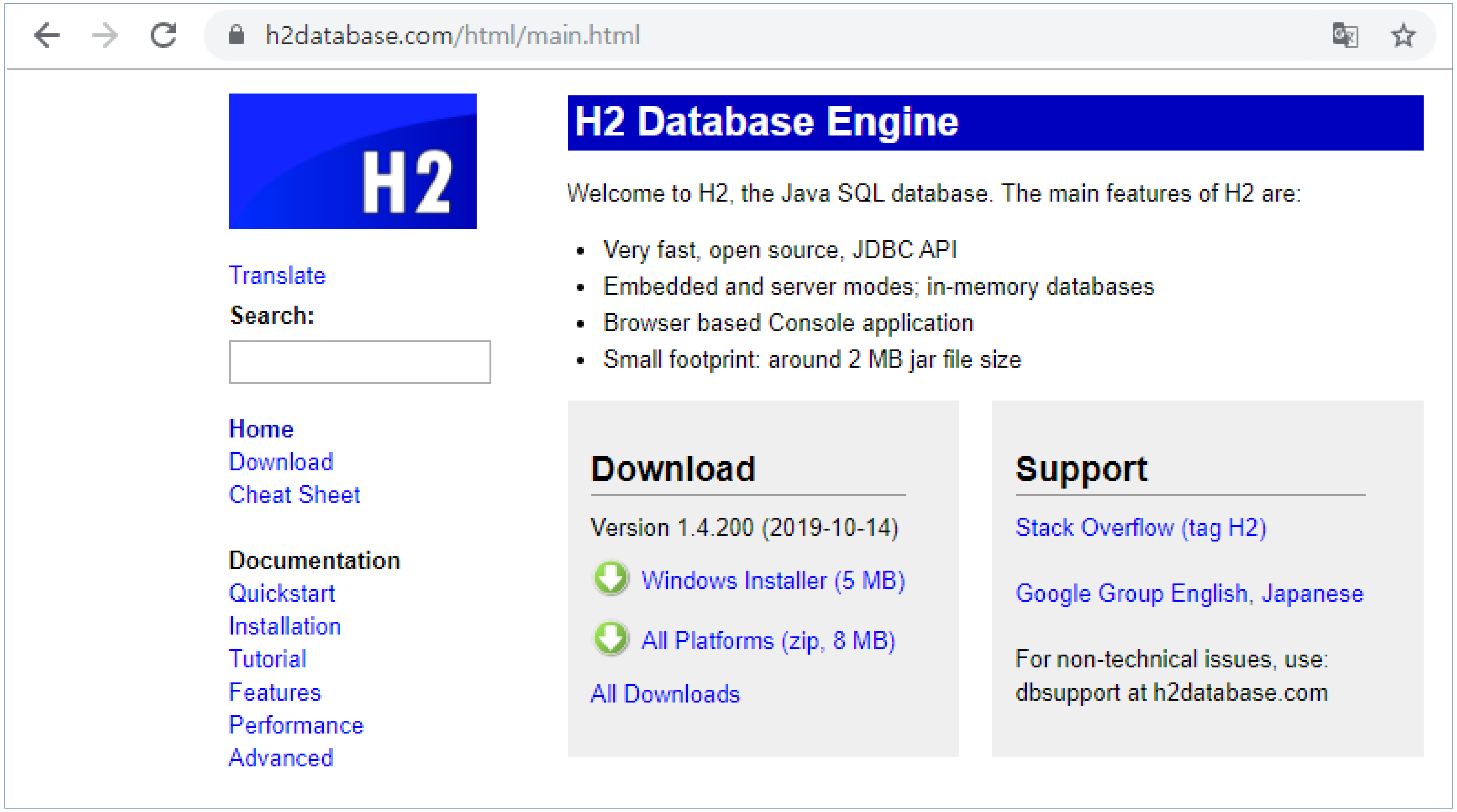
– Dto.java

```
public class MemberDto {
    private String mId;
    private String mPw;
    private String mName;
    private String creDate;
    ...
}
```

■ MyBatis

● H2 데이터베이스 설치

– https://h2database.com/html/main.html



h2database.com/html/main.html

Translate

Search:

Home
Download
Cheat Sheet

Documentation
Quickstart
Installation
Tutorial
Features
Performance
Advanced

H2 Database Engine

Welcome to H2, the Java SQL database. The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes; in-memory databases
- Browser based Console application
- Small footprint: around 2 MB jar file size

Download

Version 1.4.200 (2019-10-14)

Windows Installer (5 MB)

All Platforms (zip, 8 MB)

All Downloads

Support

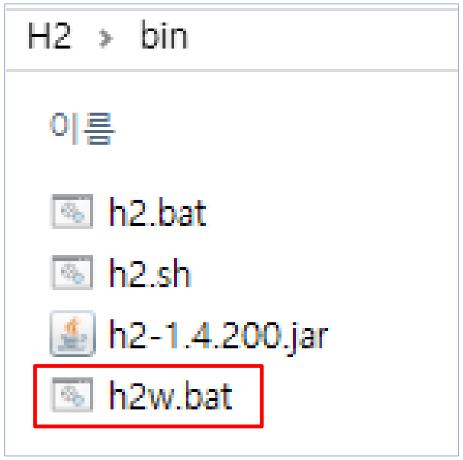
Stack Overflow (tag H2)

Google Group English, Japanese

For non-technical issues, use:
dbsupport at h2database.com

■ MyBatis

● H2 데이터베이스 실행


H2 › bin

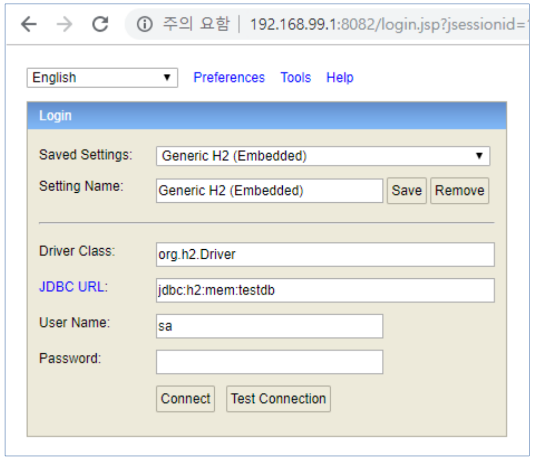
이름

h2.bat

h2.sh

h2-1.4.200.jar

h2w.bat


192.168.99.1:8082/login.jsp?jsessionid=

Saved Settings: Generic H2 (Embedded)

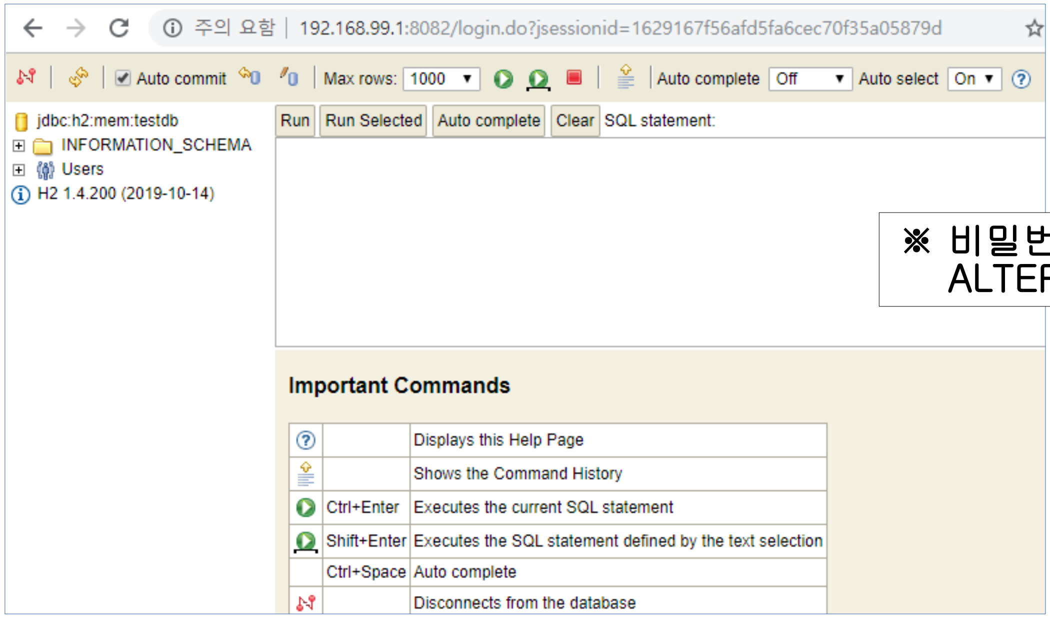
Setting Name: Generic H2 (Embedded)

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb

User Name: sa

Password:


192.168.99.1:8082/login.do?jsessionid=1629167f56afd5fa6cec70f35a05879d

Auto commit · Max rows: 1000 · Auto complete: Off · Auto select: On

jdbc:h2:mem:testdb
- INFORMATION_SCHEMA
- Users
- H2 1.4.200 (2019-10-14)

Important Commands

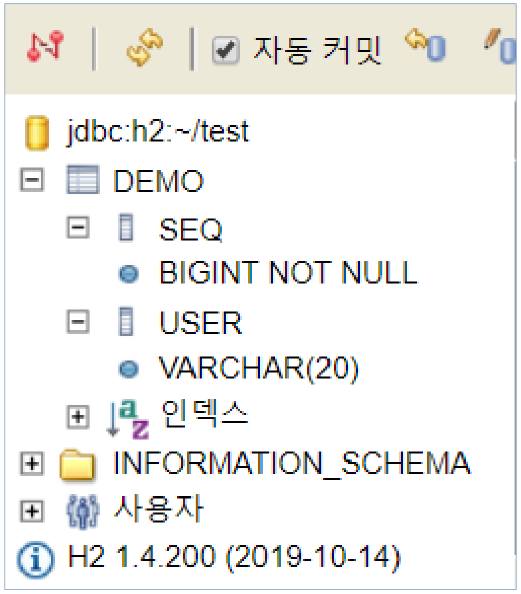
?		Displays this Help Page
		Shows the Command History
	Ctrl+Enter	Executes the current SQL statement
	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
		Disconnects from the database

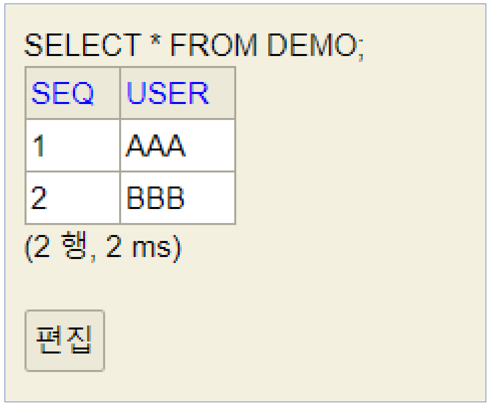
※ 비밀번호 변경
ALTER USER sa SET PASSWORD '1234'

■ MyBatis

● Table 생성 및 Data 입력

```
CREATE TABLE DEMO (
  SEQ BIGINT PRIMARY KEY,
  USER VARCHAR(20)
);

INSERT INTO DEMO (SEQ, USER) VALUES (1, 'AAA');
INSERT INTO DEMO (SEQ, USER) VALUES (2, 'BBB');

SELECT * FROM DEMO;
```





SELECT * FROM DEMO;

SEQ	USER
1	AAA
2	BBB

(2 행, 2 ms)

■ MyBatis

● build.gradle

```
plugins {
	id 'org.springframework.boot' version '2.2.1.RELEASE'
	id 'io.spring.dependency-management' version '1.0.8.RELEASE'
	id 'java'
}

group = 'com.ggoreb'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

…

Dependencies {

	…
	runtimeOnly 'com.h2database:h2'

	…
	 implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.1.1'
}

test {
	useJUnitPlatform()
}
```

– Gradle → Refresh Gradle Project

■ MyBatis

● application.properties

```
# log level
logging.level.com.ggoreb.basic=trace

# datasource
spring.datasource.url=jdbc:h2:~/test
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# mybatis
mybatis.mapper-locations=classpath:mapper/**/*.xml

# jpa
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.show-sql=true
```

■ MyBatis

- model/Demo.java

```java
package com.ggoreb.basic.model;

import lombok.Data;

@Data
public class Demo {
	private long seq;
	private String user;
}
```

- mapper/DemoMapper.java

```java
package com.ggoreb.basic.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Mapper;

import com.ggoreb.basic.model.Demo;

@Mapper
public interface DemoMapper {
	public List<Demo> getDemoList();
}
```

■ MyBatis

● config/MyBatisConfig.java

```java
package com.ggoreb.basic.config;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@MapperScan(basePackages = "com.ggoreb.basic.mapper")
public class MyBatisConfig {

}
```

● resources/mapper/demo.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.ggoreb.basic.mapper.DemoMapper">
    <select id="getDemoList" resultType="com.ggoreb.basic.model.Demo">
        select seq, user from demo
    </select>
</mapper>
```

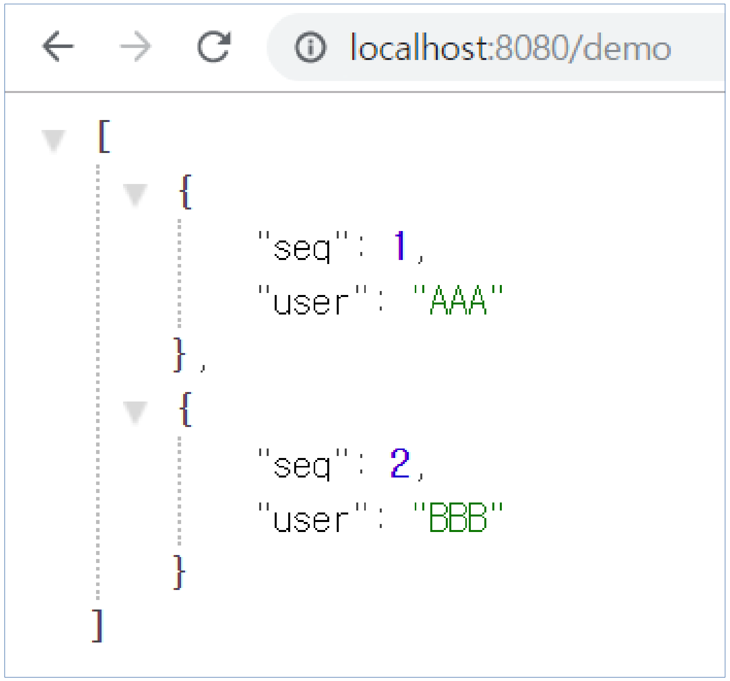
■ MyBatis

- controller/MyBatisController.java

```java
package com.ggoreb.basic.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ggoreb.basic.mapper.DemoMapper;
import com.ggoreb.basic.model.Demo;

@RestController
public class MyBatisController {
	@Autowired
	DemoMapper demoMapper;

	@GetMapping("/demo")
	public List<Demo> demo() {
		List<Demo> list = demoMapper.getDemoList();
		return list;
	}
}
```

■ MyBatis

● http://localhost:8080/demo



localhost:8080/demo

```
[
    {
        "seq": 1,
        "user": "AAA"
    },
    {
        "seq": 2,
        "user": "BBB"
    }
]
```

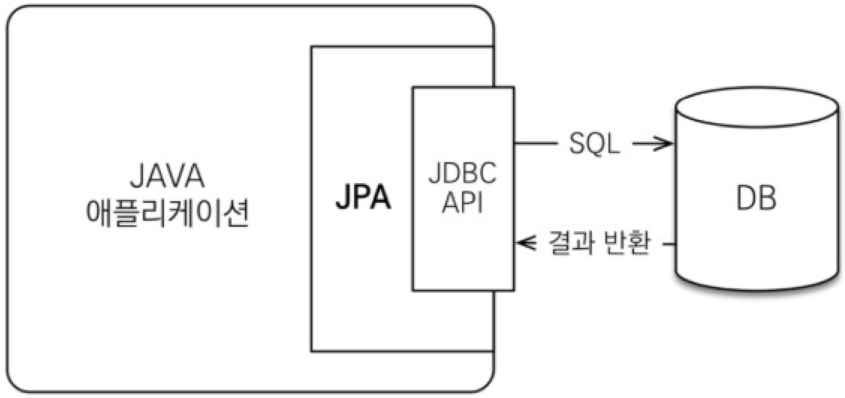
■ JPA

● Java Persistence API

● ORM 프레임워크 (Object Relational Mapping)
  - 객체는 객체대로, 관계형 데이터베이스는 관계형 데이터베이스대로 설계

● 특징
  - DAO와 Database Table의 강한 의존성 문제 해결
  - Model(자바 클래스)을 작성하면 자동으로 Table 생성
  - SQL 문장을 이용하지 않고 메소드를 호출하면 자동으로 SQL 문장 실행

● 장점
  - 생산성 향상
  - 유지보수
  - 특정 벤더(DB)에 종속적이지 않음



■ JPA

- application.properties

```
…

# jpa
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.show-sql=true
```

- model/Product.java

```java
package com.ggoreb.basic.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.Data;

@Data
@Entity
public class Product {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private long id;
	private String name;
	private int price;
	private int count;
}
```

■ JPA

● repository/ProductRepository.java

```java
package com.ggoreb.basic.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.ggoreb.basic.model.Product;

@Repository
public interface ProductRepository extends JpaRepository<Product, Long>{

}
```

● 주요 메소드

– 데이터 입력 및 수정 (insert / update) : save(T t)

– 데이터 조회 (select) : findAll() / findById(ID)

– 데이터 삭제 (delete) : delete(T t)

– 데이터 개수 확인 : count()

■ JPA

● 사용자 정의 메소드 규칙

키워드	메소드명	키워드	메소드명
And	findByLastnameAndFirstname	Like	findByFirstnameLike
Or	findByLastnameOrFirstname	NotLike	findByFirstnameNotLike
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	StartingWith	findByFirstnameStartingWith
Between	findByStartDateBetween	EndingWith	findByFirstnameEndingWith
LessThan	findByAgeLessThan	Containing	findByFirstnameContaining
LessThan Equal	findByAgeLessThanEqual	OrderBy	findByAgeOrderByLastNameDesc
GreaterThan	findByAgeGreaterThan	Not	findByLastnameNot
GreaterThan Equal	findByAgeGreaterThanEqual	In	findByAgeIn(Collection<Age> ages)
After	findByStartDateAfter	NotIn	findByAgeNotIn(Collection<Age> age)
Before	findByStartDateBefore	True	findByActiveTrue()
IsNull	findByAgeIsNull	False	findByActiveFalse()
IsNotNull, NotNull	findByAge(Is)NotNull	IgnoreCase	findByFirstnameIgnoreCase

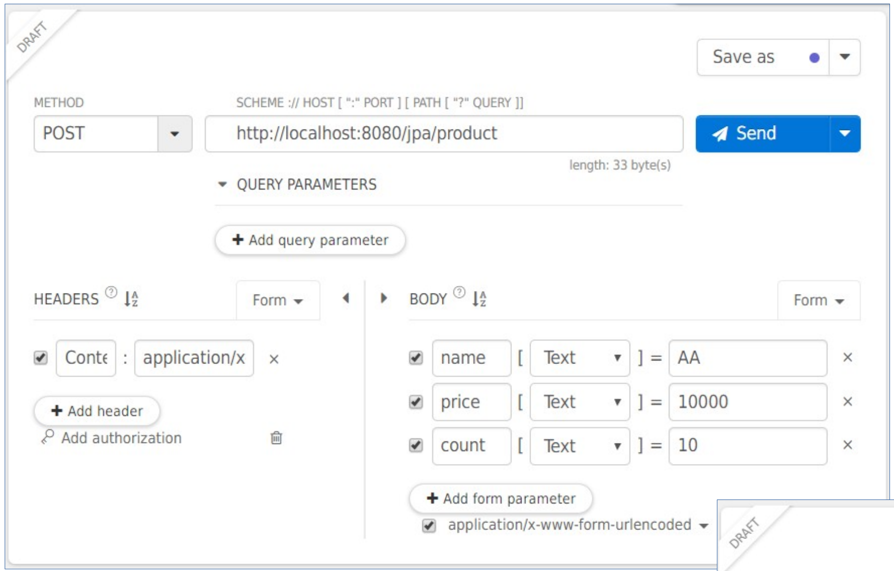
■ JPA

- controller/JpaController.java

```java
package com.ggoreb.basic.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ggoreb.basic.model.Product;
import com.ggoreb.basic.repository.ProductRepository;

@RestController
public class JpaController {
	@Autowired
	ProductRepository productRepository;

	@GetMapping("/jpa/product")
	public List<Product> product() {
		List<Product> list = productRepository.findAll();
		return list;
	}

	@PostMapping("/jpa/product")
	public String productPost(@ModelAttribute Product product) {
		productRepository.save(product);
		return "redirect:/jpa/product";
	}
}
```

■ JPA

- http://localhost:8080/jpa/product (Restlet client – POST 요청)

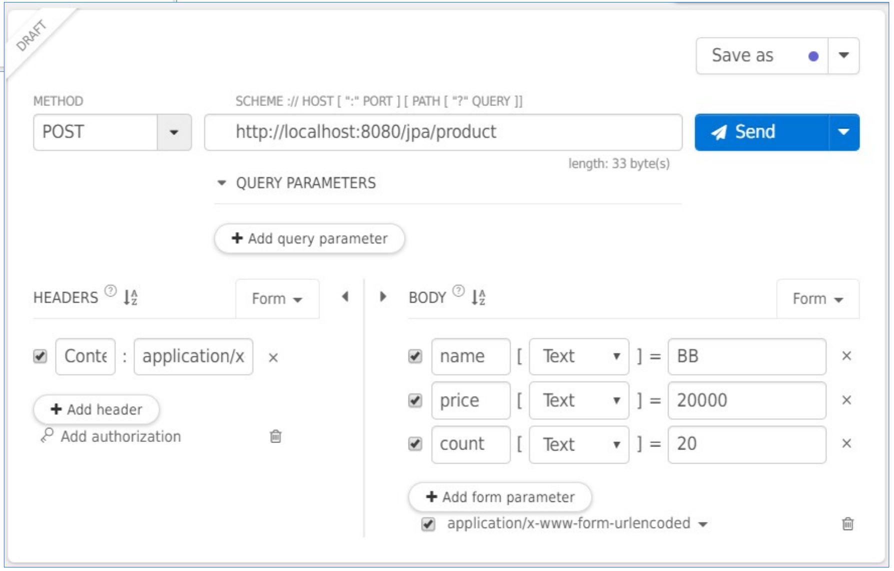


METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8080/jpa/product
length: 33 byte(s)

HEADERS: Conte : application/x

name	Text	AA
price	Text	10000
count	Text	10

application/x-www-form-urlencoded

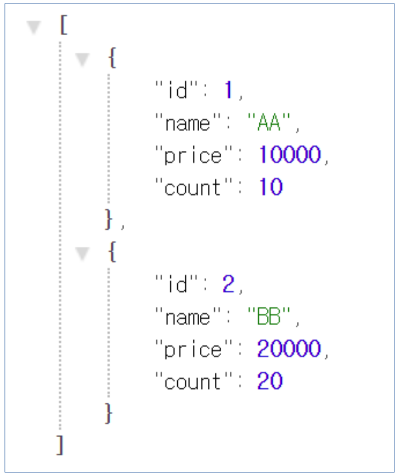


METHOD: POST
SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]: http://localhost:8080/jpa/product
length: 33 byte(s)

HEADERS: Conte : application/x

name	Text	BB
price	Text	20000
count	Text	20

application/x-www-form-urlencoded



```
[
    {
        "id": 1,
        "name": "AA",
        "price": 10000,
        "count": 10
    },
    {
        "id": 2,
        "name": "BB",
        "price": 20000,
        "count": 20
    }
]
```

■ File Upload

- controller/UploadController.java – MultipartHttpServletRequest

```java
package com.ggoreb.basic.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.multipart.MultipartHttpServletRequest;

@Controller
public class UploadController {
	@GetMapping("/upload1")
	public String upload1() {
		return "upload1";
	}

	@PostMapping("/upload1")
	@ResponseBody
	public String upload1Post(MultipartHttpServletRequest mRequest) {
		String result = "";

		MultipartFile mFile = mRequest.getFile("file");
		String oName = mFile.getOriginalFilename();
		result += oName + "\n";

		return result;
	}
}
```

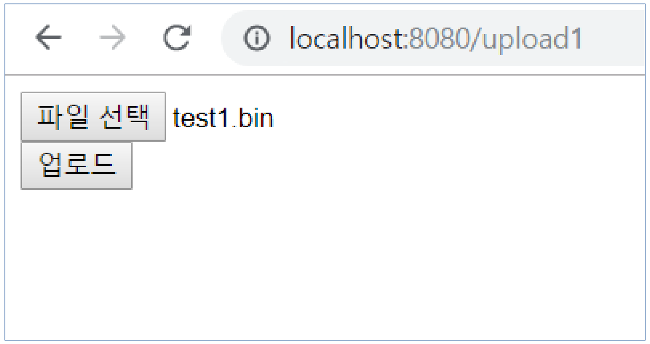
■ File Upload

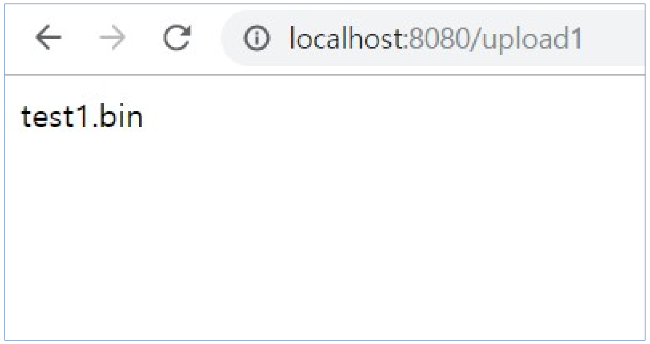
- templates/upload1.html

```
<form method="post" enctype="multipart/form-data">
  <input type="file" name="file" multiple><br>
  <input type="submit" value="업로드">
</form>
```

- application.properties

```

…

# file upload
spring.servlet.multipart.max-file-size=2097152
spring.servlet.multipart.max-request-size=2097152
```


localhost:8080/upload1


localhost:8080/upload1

test1.bin

■ File Upload

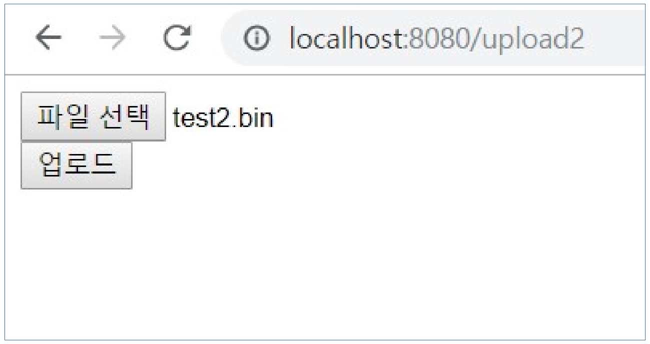
● controller/UploadController.java – @RequestParam

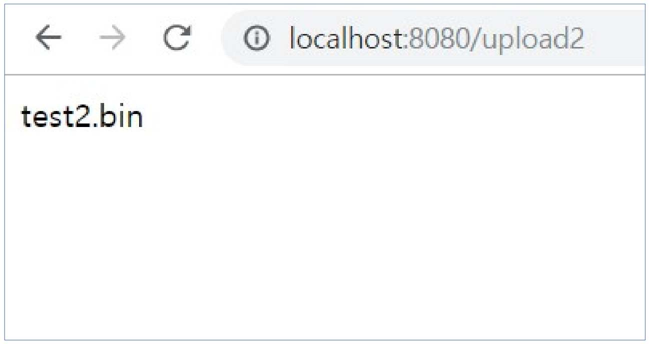
```
…

@GetMapping("/upload2")
public String upload2() {
	return "upload2";
}

@PostMapping("/upload2")
@ResponseBody
public String upload2Post(@RequestParam("file") MultipartFile mFile) {
	String result = "";

	String oName = mFile.getOriginalFilename();
	result += oName + "\n";

	return result;
}
```

■ File Upload

● templates/upload2.html

```
<form method="post" enctype="multipart/form-data">
  <input type="file" name="file" multiple><br>
  <input type="submit" value="업로드">
</form>
```


localhost:8080/upload2


localhost:8080/upload2

test2.bin

■ File Upload

● controller/UploadController.java – @ModelAttribute

```
     …

     @GetMapping("/upload3")
     public String upload3() {
          return "upload3";
     }

     @PostMapping("/upload3")
     @ResponseBody
     public String upload3Post(@ModelAttribute FileInfo info) {
          String result = "";

          String oName = info.getFile().getOriginalFilename();
          result += oName + "\n";

          return result;
     }
```

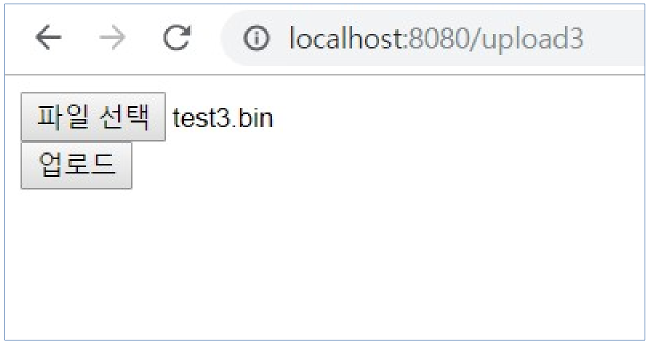
● model/FileInfo.java

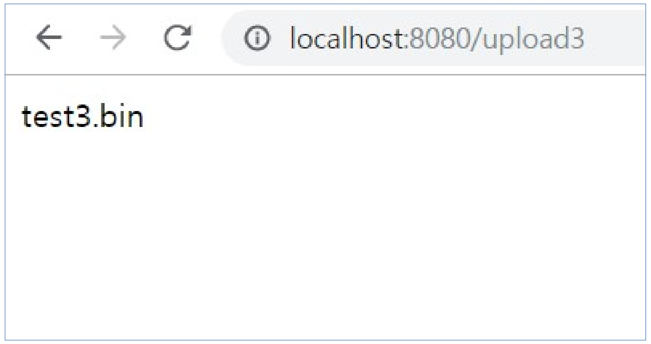
```
package com.ggoreb.basic.model;

import org.springframework.web.multipart.MultipartFile;

import lombok.Data;

@Data
public class FileInfo {
     private MultipartFile file;
}
```

■ File Upload

● templates/upload3.html

```
<form method="post" enctype="multipart/form-data">
  <input type="file" name="file" multiple><br>
  <input type="submit" value="업로드">
</form>
```





test3.bin

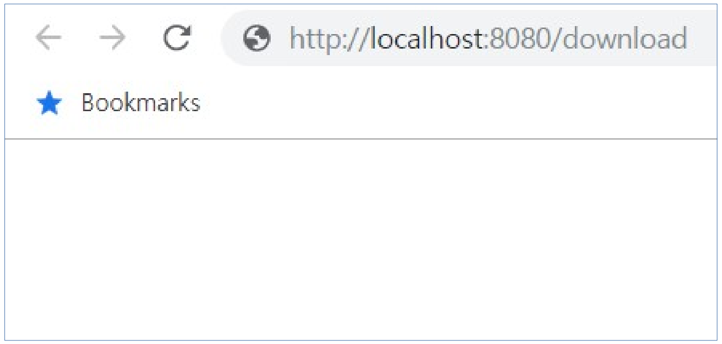
■ File Download

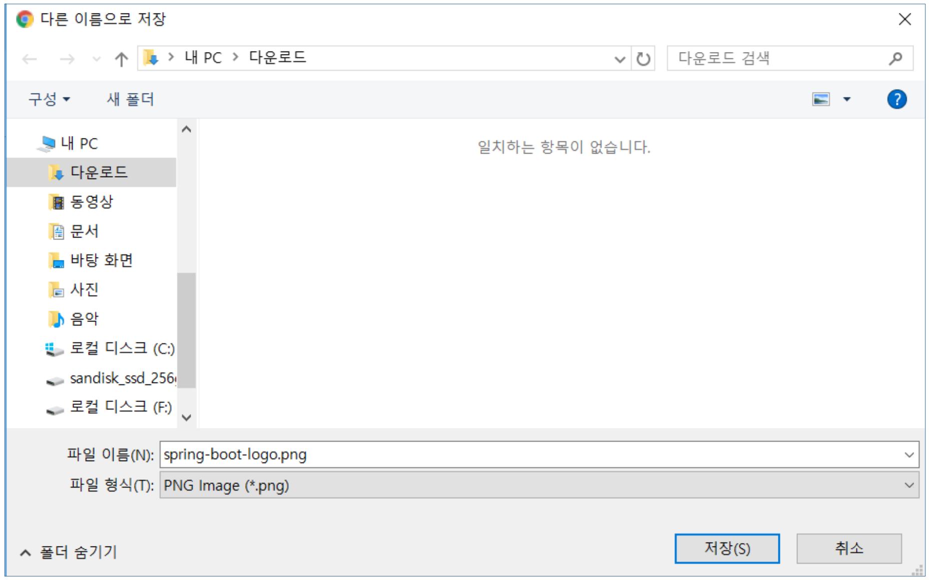
- controller/DownloadController.java

```java
package com.ggoreb.basic.controller;

import java.io.File;
import java.io.FileInputStream;
import java.net.URLEncoder;

import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.Resource;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class DownloadController {
  @GetMapping("/download")
  public ResponseEntity<Resource> download() throws Exception {
    File file = new File("f:/spring-boot-logo.png");
    InputStreamResource resource = new InputStreamResource(new FileInputStream(file));

    return ResponseEntity.ok()
            .header("content-disposition",
                    "filename=" + URLEncoder.encode(file.getName(), "utf-8"))
            .contentLength(file.length())
            .contentType(MediaType.parseMediaType("application/octet-stream"))
            .body(resource);
  }
}
```

■ File Download

- http://localhost:8080/download





파일 이름(N): spring-boot-logo.png
파일 형식(T): PNG Image (*.png)

■ RestTemplate

- HTTP 통신에 유용하게 사용할 수 있는 라이브러리
- 기계적이고 반복적인 코드를 최대한 줄여줌
- JSON / XML 형식의 응답결과에 대해 처리 지원
- 주요 메소드

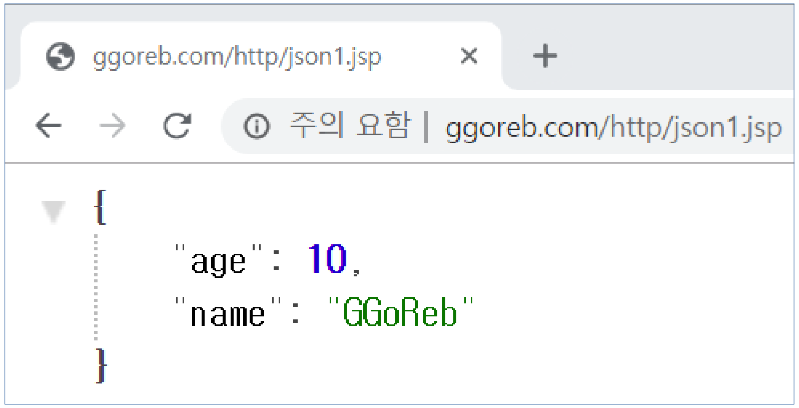
메소드명	HTTP 메소드	설명
getForObject	GET	GET 요청 후 지정 Object 형태로 응답 처리
getForEntity	GET	GET 요청 후 ResponseEntity로 응답 처리
postForObject	POST	POST 요청 후 지정 Object 형태로 응답 처리
postForEntity	POST	POST 요청 후 ResponseEntity로 응답 처리
put	PUT	PUT 요청
delete	DELETE	DELETE 요청
exchange	ALL	HTTP 헤더를 지정하여 요청 및 응답 처리

■ RestTemplate

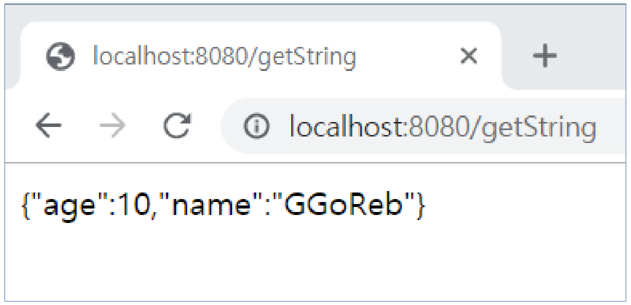
● controller/RestTemplateController.java

– getForObject / String

```
package com.ggoreb.basic.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class RestTemplateController {
  @GetMapping("/getString")
  public String getString() {
    RestTemplate rt = new RestTemplate();
    String result = rt.getForObject("http://ggoreb.com/http/json1.jsp", String.class);
    return result;
  }
}
```


ggoreb.com/http/json1.jsp
주의 요함 | ggoreb.com/http/json1.jsp

```
{
    "age": 10,
    "name": "GGoReb"
}
```



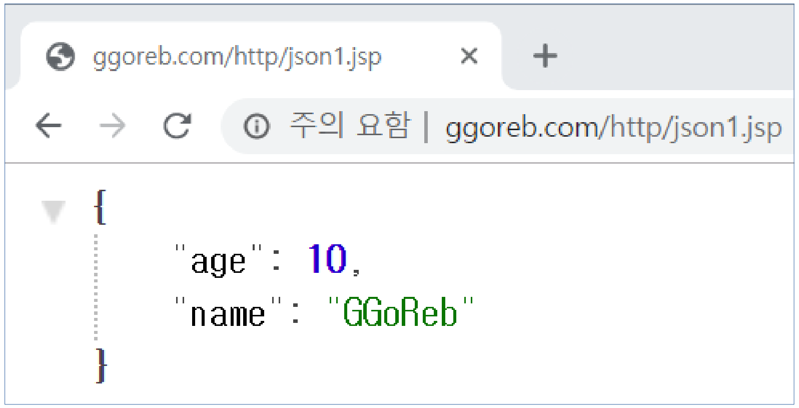
```
{"age":10,"name":"GGoReb"}
```

■ RestTemplate

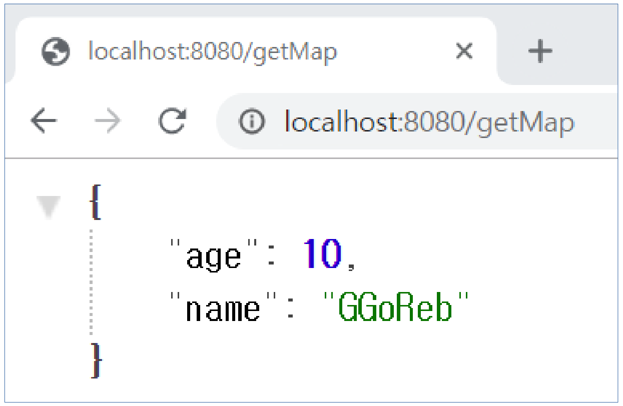
● controller/RestTemplateController.java

– getForObject / Map

```
@GetMapping("/getMap")
public Map<String, Object> getMap() {
  RestTemplate rt = new RestTemplate();
  Map<String, Object> map =
      rt.getForObject("http://ggoreb.com/http/json1.jsp", Map.class);
  return map;
}
```


ggoreb.com/http/json1.jsp
주의 요함 | ggoreb.com/http/json1.jsp

```
{
    "age": 10,
    "name": "GGoReb"
}
```

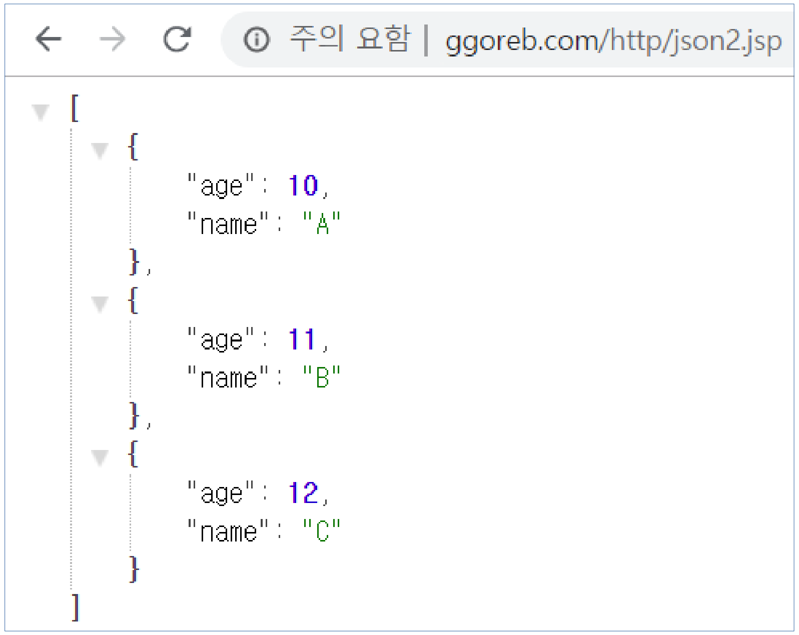


```
{
    "age": 10,
    "name": "GGoReb"
}
```

■ RestTemplate

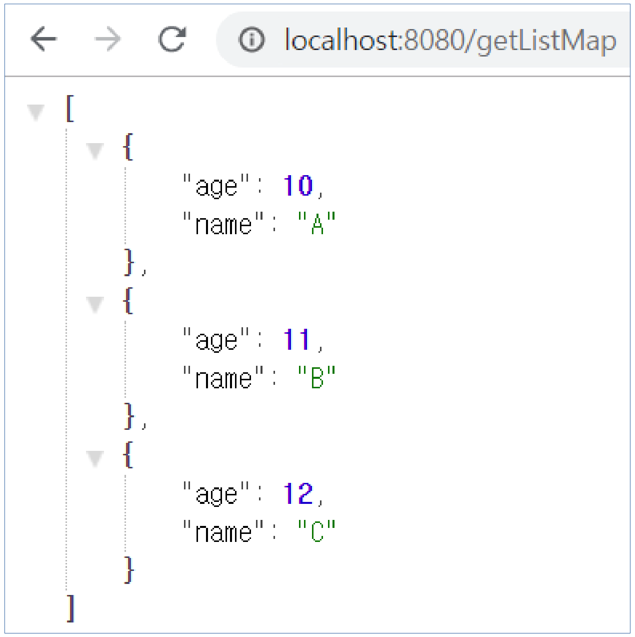
- controller/RestTemplateController.java
  - getForObject / List<Map>

```java
@GetMapping("/getListMap")
public List<Map<String, Object>> getListMap() {
  RestTemplate rt = new RestTemplate();
  List<Map<String, Object>> list =
      rt.getForObject("http://ggoreb.com/http/json2.jsp", List.class);
  return list;
}
```



주의 요함 | ggoreb.com/http/json2.jsp

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```



localhost:8080/getListMap

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```

■ RestTemplate

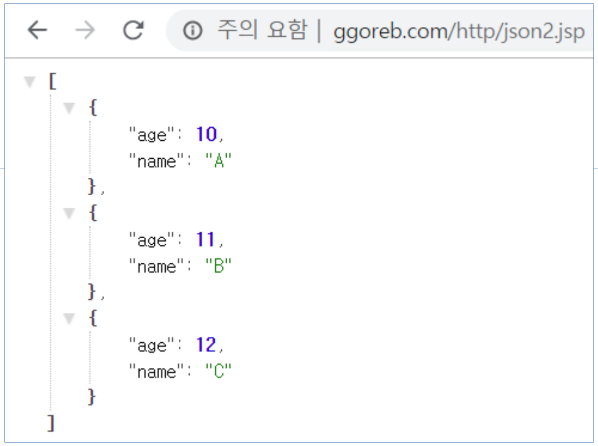
● controller/RestTemplateController.java

– getForObject / List<Object>

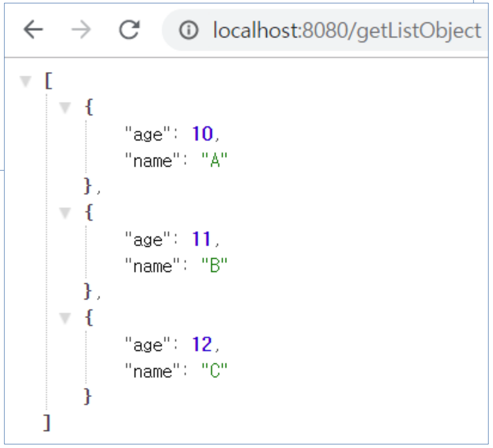
```
  @GetMapping("/getListObject")
  public List<JsonData> getListObject() {
    RestTemplate rt = new RestTemplate();
    List<JsonData> list =
        rt.getForObject("http://ggoreb.com/http/json2.jsp", List.class);
    return list;
  }
```

● model/JsonData.java

```
package com.ggoreb.basic.model;

import lombok.Data;

@Data
public class JsonData {
  private int age;
  private String name;
}
```


주의 요함 | ggoreb.com/http/json2.jsp

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```


localhost:8080/getListObject

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```

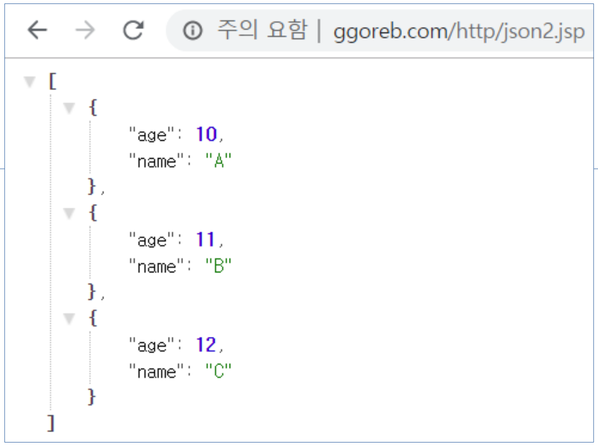
■ RestTemplate

- controller/RestTemplateController.java
  - getForObject / Object[]

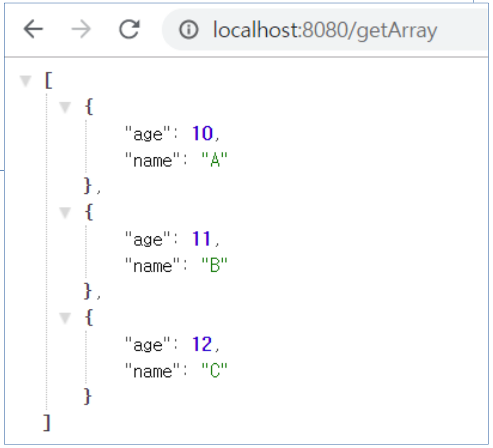
```
  @GetMapping("/getArray")
  public List<JsonData> getArray() {
    RestTemplate rt = new RestTemplate();
    JsonData[] data =
        rt.getForObject("http://ggoreb.com/http/json2.jsp", JsonData[].class);
    List<JsonData> list = Arrays.asList(data);
    return list;
  }
```

- model/JsonData.java

```
package com.ggoreb.basic.model;

import lombok.Data;

@Data
public class JsonData {
  private int age;
  private String name;
}
```


주의 요함 | ggoreb.com/http/json2.jsp

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```


localhost:8080/getArray

```
[
  {
    "age": 10,
    "name": "A"
  },
  {
    "age": 11,
    "name": "B"
  },
  {
    "age": 12,
    "name": "C"
  }
]
```

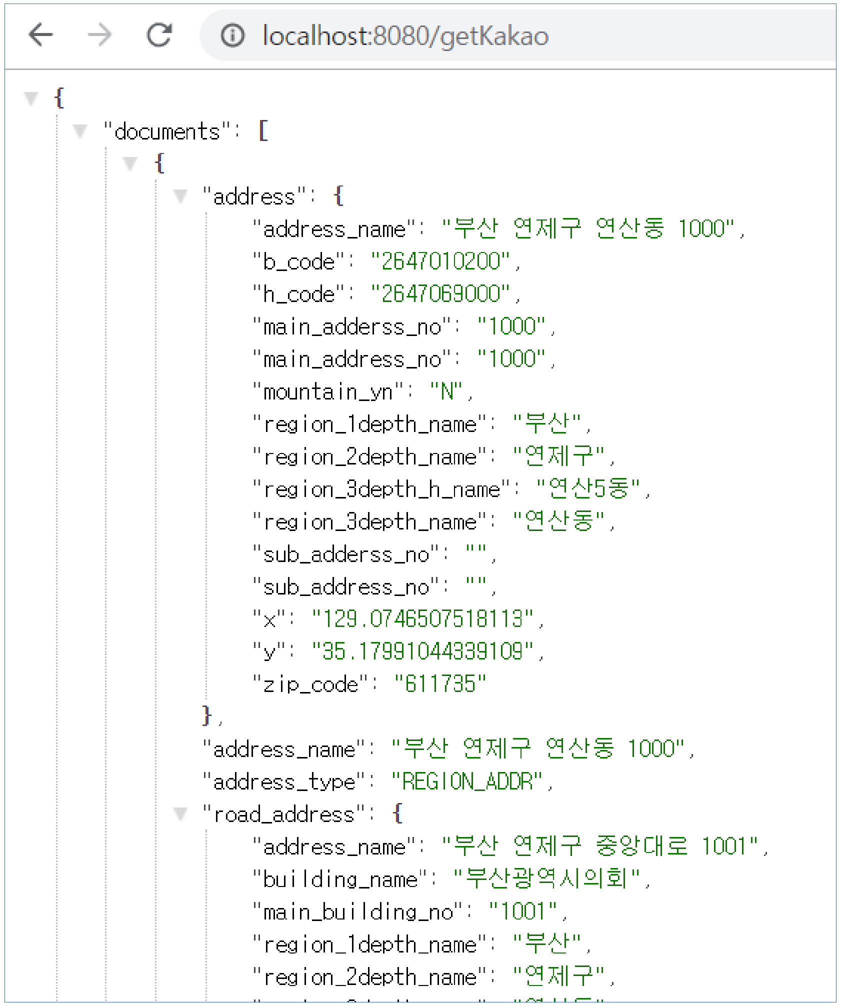
■ RestTemplate

- controller/RestTemplateController.java
  - exchange / Kakao Map API

```java
@GetMapping("/getKakao")
public ResponseEntity<Map> getKakao() {
  RestTemplate rt = new RestTemplate();
  RequestEntity requestEntity = null;

  try {

    requestEntity = RequestEntity.get(
        new URI("https://dapi.kakao.com/v2/local/search/address.json?query=" +
            URLEncoder.encode("부산 연제구 연산동 1000", "utf-8")))
        .header("Authorization", "KakaoAK d4be7b479f4b4cbd99bd19ae87f88b4b")
        .build();

  } catch (UnsupportedEncodingException e) {
    e.printStackTrace();
  } catch (URISyntaxException e) {
    e.printStackTrace();
  }

  ResponseEntity<Map> entity = rt.exchange(requestEntity, Map.class);

  return entity;
}
```

■ RestTemplate

- controller/RestTemplateController.java
  - **exchange / Kakao Map API**



localhost:8080/getKakao

```
{
  "documents": [
    {
      "address": {
        "address_name": "부산 연제구 연산동 1000",
        "b_code": "2647010200",
        "h_code": "2647069000",
        "main_adderss_no": "1000",
        "main_address_no": "1000",
        "mountain_yn": "N",
        "region_1depth_name": "부산",
        "region_2depth_name": "연제구",
        "region_3depth_h_name": "연산5동",
        "region_3depth_name": "연산동",
        "sub_adderss_no": "",
        "sub_address_no": "",
        "x": "129.0746507518113",
        "y": "35.17991044339109",
        "zip_code": "611735"
      },
      "address_name": "부산 연제구 연산동 1000",
      "address_type": "REGION_ADDR",
      "road_address": {
        "address_name": "부산 연제구 중앙대로 1001",
        "building_name": "부산광역시의회",
        "main_building_no": "1001",
        "region_1depth_name": "부산",
        "region_2depth_name": "연제구",
```

■ RestTemplate

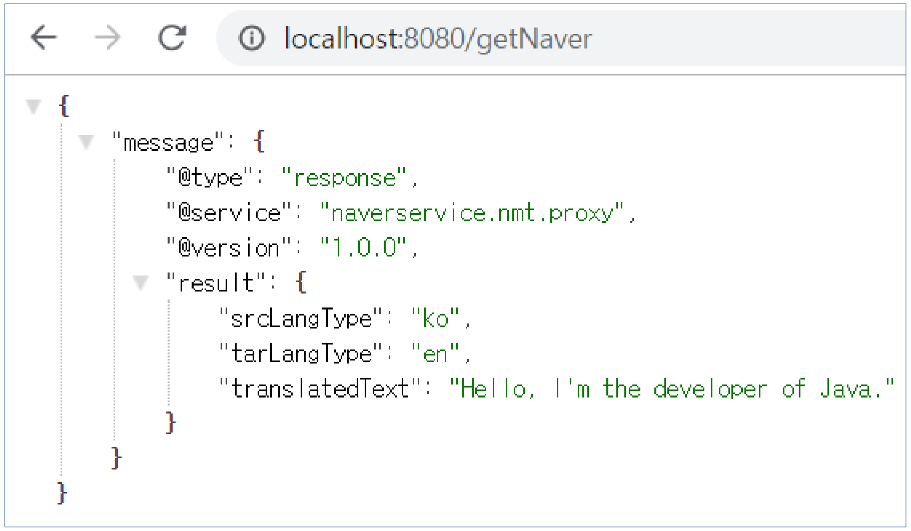
- controller/RestTemplateController.java
  - exchange / Naver Papago API

```
@GetMapping("/getNaver")
public ResponseEntity<Map> getNaver() {
  RestTemplate rt = new RestTemplate();

  RequestEntity<Map<String, String>> requestEntity = null;
  try {
    Map<String, String> body = new HashMap<>();
    body.put("source", "ko");
    body.put("target", "en");
    body.put("text", "안녕하세요. 저는 자바 개발자입니다.");

    requestEntity = RequestEntity.post(
        new URI("https://openapi.naver.com/v1/papago/n2mt"))
        .header("X-Naver-Client-Id", "OpcnSsAIn37qIu6Iyad6")
        .header("X-Naver-Client-Secret", "p7qtbsYx8N")
        .body(body);
  } catch (URISyntaxException e) {
    e.printStackTrace();
  }

  ResponseEntity<Map> entity = rt.exchange(requestEntity, Map.class);
  return entity;
}
```

■ RestTemplate

● controller/RestTemplateController.java

– **exchange / Naver Papago API**


localhost:8080/getNaver

```
{
    "message": {
        "@type": "response",
        "@service": "naverservice.nmt.proxy",
        "@version": "1.0.0",
        "result": {
            "srcLangType": "ko",
            "tarLangType": "en",
            "translatedText": "Hello, I'm the developer of Java."
        }
    }
}
```