**1. response객체에 문자열 담기**

첫번째로 살펴볼 방법은 HttpServletResponse 객체에 문자열을 담아 보내는 방법입니다. Servlet기반인 HttpServletResponse 객체를 사용하므로 Spring MVC뿐아니라 Servlet기반의 타 프레임워크에서도 사용가능하다는 장점이 있습니다. 하지만, 단순히 문자열을 사용자가 임의로 Json형식으로 작성하는 만큼 오탈자등을 고려해야하는 번거로움이 있습니다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //Script  $("#joinOk").bind("click",function(){      $.ajax({          url : contextPath+"/ajax.seo",          type: "get",          data : { "id" : $("#id").val() },          success : function(responseData){              $("#ajax").remove();              var data = JSON.parse(responseData);              if(!data){                  alert("존재하지 않는 ID입니다");                  return false;              }              var html = '';              html += '<form class="form-signin" action="" id="ajax">';              html += '이름<input type="text" class="form-control"  name="name" value="'+data.name+'">';              html += '아이디<input type="text" class="form-control" name=id" value="'+data.id+'">';              html += '이메일<input type="text" class="form-control"  name="email" value="'+data.email+'">';              html += '비밀번호<input type="text" class="form-control" name="password" value="'+data.password+'">';              html += '</form>';              $("#container").after(html);          }      });  }); |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | //Controller  @RequestMapping(value= "/ajax.seo", method=RequestMethod.GET)  public void AjaxView(          @RequestParam("id") String id,          HttpServletResponse response)  {      String personJson;        SocialPerson person = dao.getPerson(id);      if(person != null){          personJson = "{\"id\":\""+person.getId()                      +"\",\"name\":\""+person.getName()                      +"\",\"password\":\""+person.getPassword()                      +"\",\"email\":\""+person.getEmail()+"\"}";      }      else{          personJson = "null";      }      try {          response.getWriter().print(personJson);      } catch (IOException e) {          e.printStackTrace();      }  } |

위 코드는 HttpServeltResponse 객체 활용에 대한 간단한 예제 코드입니다. Script코드에서 서버(Controller)에 input태그에 입력된  id값을  전송하면  Controller에서는 해당 데이터를 parameter로 받고 그 id값으로 DB를 조회합니다. 조회 성공시 개발자가 문자열을 Json형식으로 맞추어 주며, 조회 실패시 “null”이라는 문자열을 넘겨줍니다. 그리고 response객체의 getWriter메소드를 통하여 PrintWriter 객체를 가져오고 print메소드를 사용하여 Script로 해당 문자열을 전송합니다.  
그렇게 되면 ajax success함수 responseData에 서버에서 넘겨준 문자열이 저장됩니다. responseData는 문자열 이므로 JSON.parse함수를 통하여 json객체로 파싱을 해준 후 구현합니다.

**2. ObjectMapper**

두번째로 살펴볼 방법은 ObjectMapper객체를 사용하는 방법입니다. ObjectMapper 객체를 사용하여도 HttpServletResponse 객체를 사용합니다. 하지만 ObjectMapper객체를 사용하게되면 위 방법처럼 문자열을 개발자가 작성할 필요없이 해당 객체를 Json형식의 문자열로 바꾸어주게 되어 개발자가 좀 더 간단하게 구현할 수 있습니다.  
ObjectMapper 객체를 사용 하기위해서는 구현에 앞서 jackson-databind 라이브러리를 프로젝트에 추가해야 합니다. jackson-databind라이브러리를 다운받아  lib폴더에 추가 시켜주거나 Maven 프로젝트 일경우 pom.xml경우에 아래와 같이 dependency 해주면 됩니다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>com.fasterxml.jackson.core</groupId>      <artifactId>jackson-databind</artifactId>      <version>버전</version>  </dependency> |

위의 방법대로 라이브러리를 추가하시면 ObjectMapper객체를 사용할 수 있습니다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //Script  $("#joinOk").bind("click",function(){      $.ajax({          url : contextPath+"/ajax.seo",          type: "get",          data : { "id" : $("#id").val() },          success : function(responseData){              $("#ajax").remove();              var data = JSON.parse(responseData);              if(!data){                  alert("존재하지 않는 ID입니다");                  return false;              }              var html = '';              html += '<form class="form-signin" action="" id="ajax">';              html += '이름<input type="text" class="form-control"  name="name" value="'+data.name+'">';              html += '아이디<input type="text" class="form-control" name=id" value="'+data.id+'">';              html += '이메일<input type="text" class="form-control"  name="email" value="'+data.email+'">';              html += '비밀번호<input type="text" class="form-control" name="password" value="'+data.password+'">';              html += '</form>';              $("#container").after(html);          }      });  }); |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | //Controller  @RequestMapping(value= "/ajax.seo", method=RequestMethod.GET)  public void AjaxView(          @RequestParam("id") String id,          HttpServletResponse response)  {      ObjectMapper mapper = new ObjectMapper();        SocialPerson person = dao.getPerson(id);      try {          response.getWriter().print(mapper.writeValueAsString(person));      } catch (IOException e) {          e.printStackTrace();      }  } |

위 코드는 ObjcetMapper객체 활용의 간단한 예제코드입니다. Script코드에서 서버(Controller)에 input태그에 입력된  id값을  전송하면  Controller에서는 해당 데이터를 parameter로 받고 그 id값으로 DB를 조회합니다. ObjcetMapper객체의 writeValueAsString메소드를 사용하여 Person객체를 Json형식의 문자열로 만듭니다. 그리고 이후에 위의 방법과 마찬가지로 response객체를 활용하여 문자열을 ajax success함수로 전송합니다. responseData는 서버에서 전송된 Json형식의 문자열 이므로 JSON.parse 함수를 통하여 Json객체로 파싱을 해준 후 구현합니다.

**3. @ResponseBody**

세번째로 살펴볼 방법은 ResponseBody 어노테이션을 사용하는 방법입니다. 메소드의 return형 앞에 @ResponseBody를 붙여서 사용하게되면 해당객체가 자동으로 Json객체로 변환되어 반환됩니다. 구현에 앞서 @ResponseBody 환경을 설정하는 방법을 알아봅니다.

첫째, jackson-mapper-asl 라이브러리를 다운로드하여 lib폴더에 추가시켜주거나  Maven 프로젝트 일경우 pom.xml경우에 아래와 같이 dependency 해주면 됩니다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>         <groupId>org.codehaus.jackson</groupId>         <artifactId>jackson-mapper-asl</artifactId>         <version>버전</version>  </dependency> |

둘째, servlet xml에 아래와 같이 어노테이션을 설정해줍니다.



|  |  |
| --- | --- |
| 1 | <annotation-driven /> |

위에서 @ResponseBody방식을 사용하 기 위한 환경 설정법에 대해서 알아보았습니다.  아래는 @ResponseBody방식의 실제 Script와 Java Controller 구현 코드입니다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | //Script  $("#joinOk").bind("click",function(){          $.ajax({              url : contextPath+"/ajax.seo",              type: "get",              data : { "id" : $("#id").val() },              success : function(data){                  $("#ajax").remove();                  alert(data);                  if(!data){                      alert("존재하지 않는 ID입니다");                      return false;                  }                  var html = '';                  html += '<form class="form-signin" action="" id="ajax">';                  html += '이름<input type="text" class="form-control"  name="name" value="'+data.name+'">';                  html += '아이디<input type="text" class="form-control" name=id" value="'+data.id+'">';                  html += '이메일<input type="text" class="form-control"  name="email" value="'+data.email+'">';                  html += '비밀번호<input type="text" class="form-control" name="password" value="'+data.password+'">';                  html += '</form>';                  $("#container").after(html);              }          });        }); |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | //Controller  @RequestMapping(value= "/ajax.seo", method=RequestMethod.GET)  public @ResponseBody SocialPerson AjaxView(          @RequestParam("id") String id)  {        SocialPerson person = dao.getPerson(id);      return person;  } |

위 코드는 @ResponseBody방식에 대한 간단한 예제 코드입니다. Script코드에서 서버(Controller)에 input태그에 입력된  id값을  전송하면  Controller에서는 해당 데이터를 parameter로 받고 그 id값으로 DB를 조회합니다.  앞서 말하였듯 return형 앞에 @ResponseBody를 사용하고 해당 객체를 return해주기만 하면 ajax success함수의 data에 person객체가 Json객체로 변환 후 전송되어 파싱이 필요없습니다..

**4. jsonView**

마지막으로 살펴볼 방법은 jsonView를 사용 하는 방법입니다. 구현에 앞서 환경 설정을 해주어야 하는데, 설정방법은 다음과 같습니다.

첫째, json-lib-ext-spring 라이브러리를 다운받아 lib 폴더에 추가 시켜주거나 Maven 프로젝트 일경우 pom.xml경우에 아래와 같이 dependency 해주면 됩니다.



|  |  |
| --- | --- |
| 1  2  3  4  5 | <dependency>      <groupId>net.sf.json-lib</groupId>      <artifactId>json-lib-ext-spring</artifactId>      <version>버전</version>  </dependency> |

둘째, servlet xml에 아래와 같이  bean객체를 설정 후 viewResolver를 설정 해줍니다.



|  |  |
| --- | --- |
| 1  2  3  4 | <beans:bean id="jsonView" class="net.sf.json.spring.web.servlet.view.JsonView"/>  <beans:bean id="viewResolver" class="org.springframework.web.servlet.view.BeanNameViewResolver">      <beans:property name="order" value="1"></beans:property>  </beans:bean> |

위에서 jsonView방식을 사용하 기 위한 환경 설정법에 대해서 알아보았습니다.  아래는 jsonView방식의 실제 Script와 Java Controller 구현 코드입니다.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | // Script  $("#joinOk").bind("click",function(){          $.ajax({              url : contextPath+"/ajax.seo",              type: "get",              data : { "id" : $("#id").val() },              success : function(responseData){                  var data = responseData.person;                  $("#ajax").remove();                  if(!data){                      alert("존재하지 않는 ID입니다");                      return false;                  }                  var html = '';                  html += '<form class="form-signin" action="" id="ajax">';                  html += '이름<input type="text" class="form-control"  name="name" value="'+data.name+'">';                  html += '아이디<input type="text" class="form-control" name=id" value="'+data.id+'">';                  html += '이메일<input type="text" class="form-control"  name="email" value="'+data.email+'">';                  html += '비밀번호<input type="text" class="form-control" name="password" value="'+data.password+'">';                  html += '</form>';                  $("#container").after(html);              }          });        }); |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | //Controller  @RequestMapping(value= "/ajax.seo", method=RequestMethod.GET)  public ModelAndView AjaxView( @RequestParam("id") String id)  {      ModelAndView mav= new ModelAndView();        SocialPerson person = dao.getPerson(id);      mav.addObject("person",person);      mav.setViewName("jsonView");      return mav;  } |

위 코드는 jsonView방식에 대한 간단한 예제 코드입니다. Script코드에서 서버(Controller)에 input태그에 입력된  id값을  전송하면  Controller에서는 해당 데이터를 parameter로 받고 그 id값으로 DB를 조회합니다. 조회결과가 있으면 해당 회원(Person) 객체를 ModelandView객체에 addObject메소드를 사용하여 넣어주고, 조회결과가 없으면 null이 됩니다. 그리고 마지막으로 view명을 “jsonView”로 하면 ajax success함수 responseData에  ModelandView 객체가 Json객체로 파싱되어서 넘어옵니다. 즉, ModelandView 객체에 맵 형식(키,밸류)으로 Person객체를 입력하였으므로 responseData가 아닌 responseData.person이 Person객체가 됩니다. 넘어온 Person 객체는 json형식이기 때문에 속성 값은 data.id, data.name 등으로 구현이 됩니다.