

Spring MVC

brief tutorial

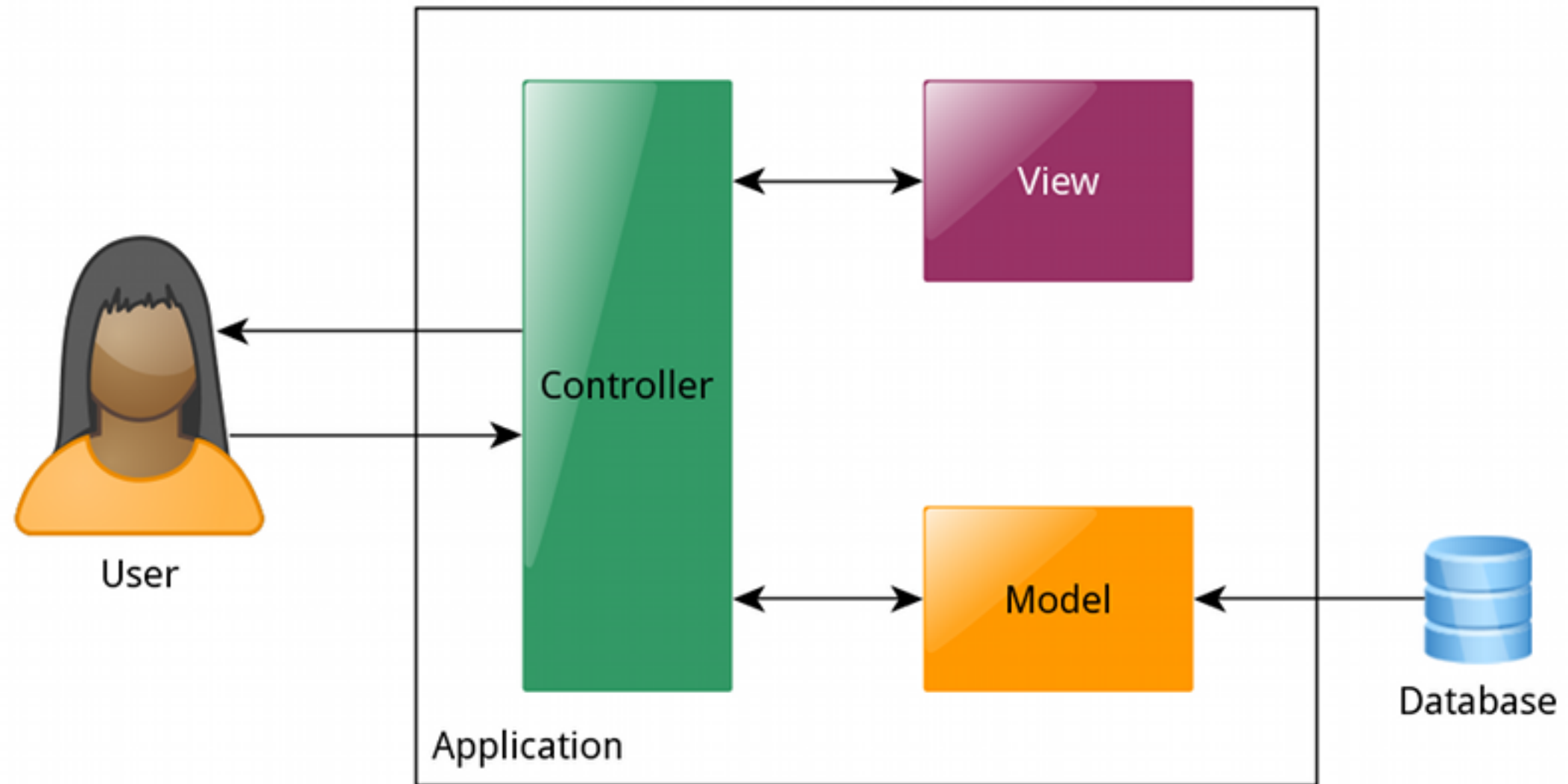
Joel Niklaus

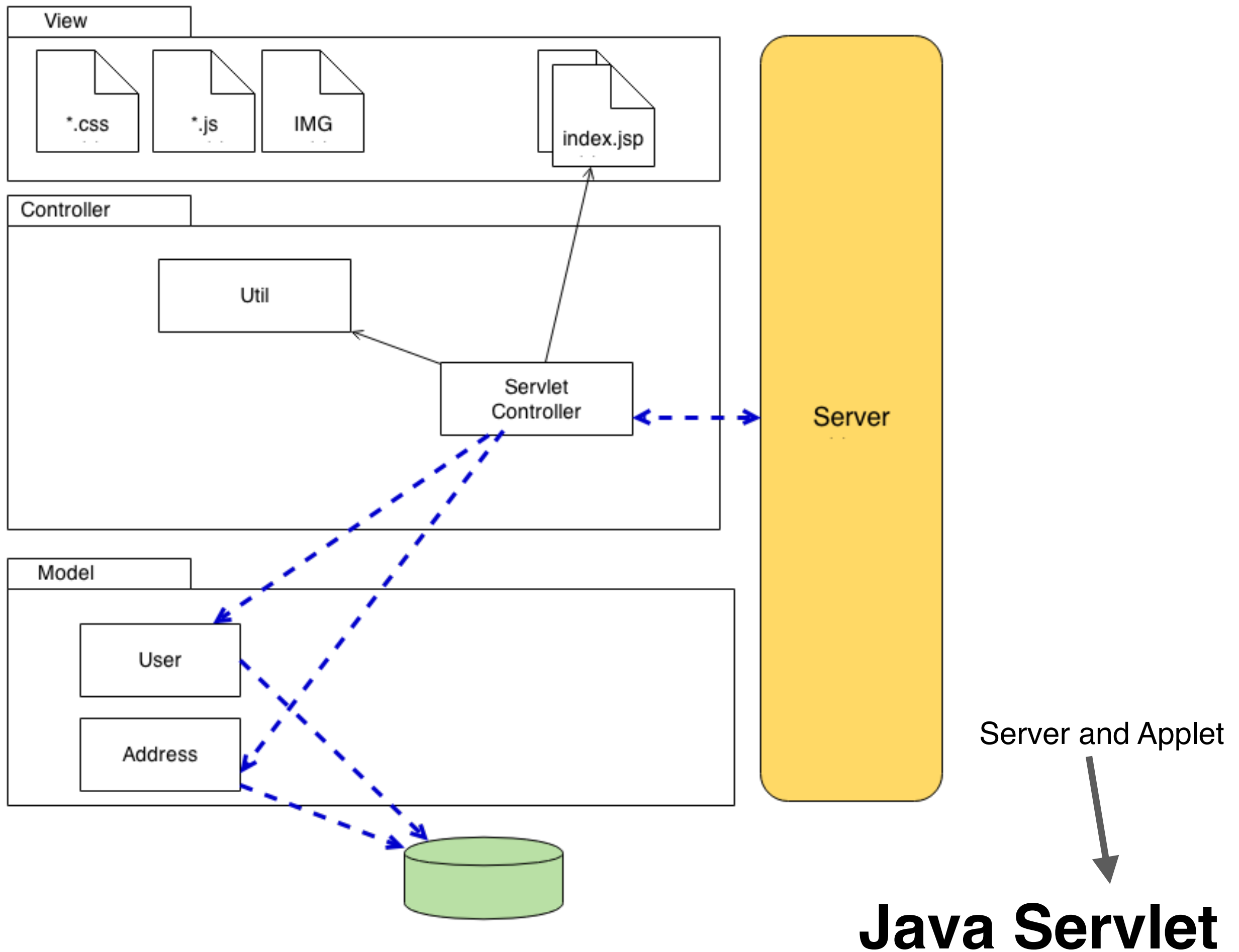
<http://www.joelniklaus.ch>



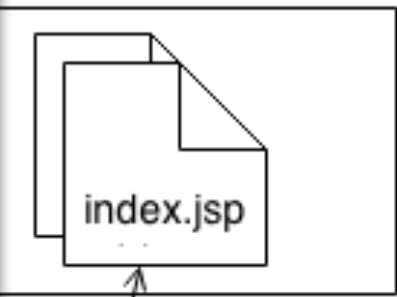
^b
**UNIVERSITÄT
BERN**

MVC

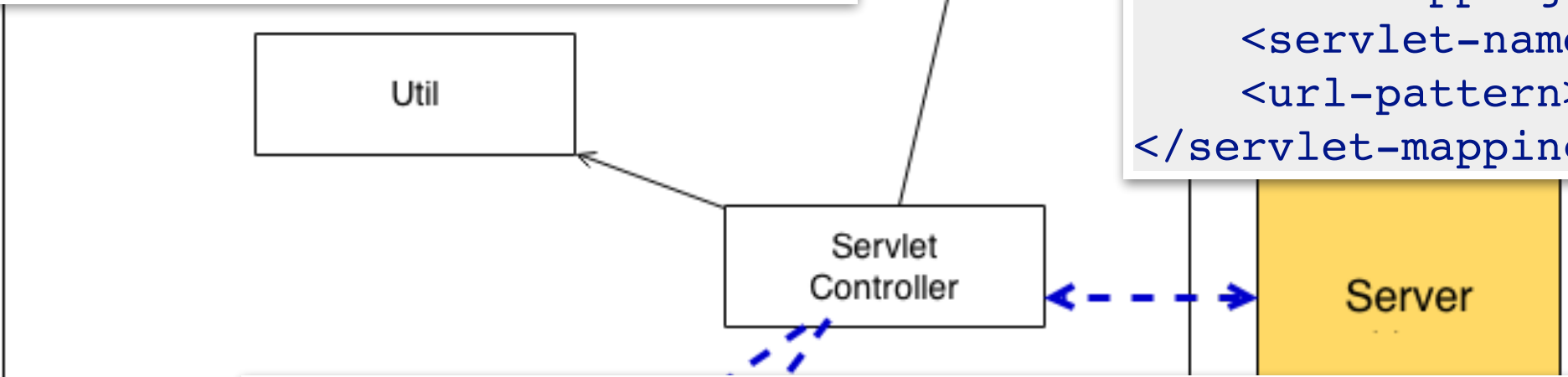




```
<%  
List styles = req.getAttr("styles");  
Iterator it = styles.iterator();  
while(it.hasNext()) {  
    out.print("<br>" + it.next());  
}  
%>
```



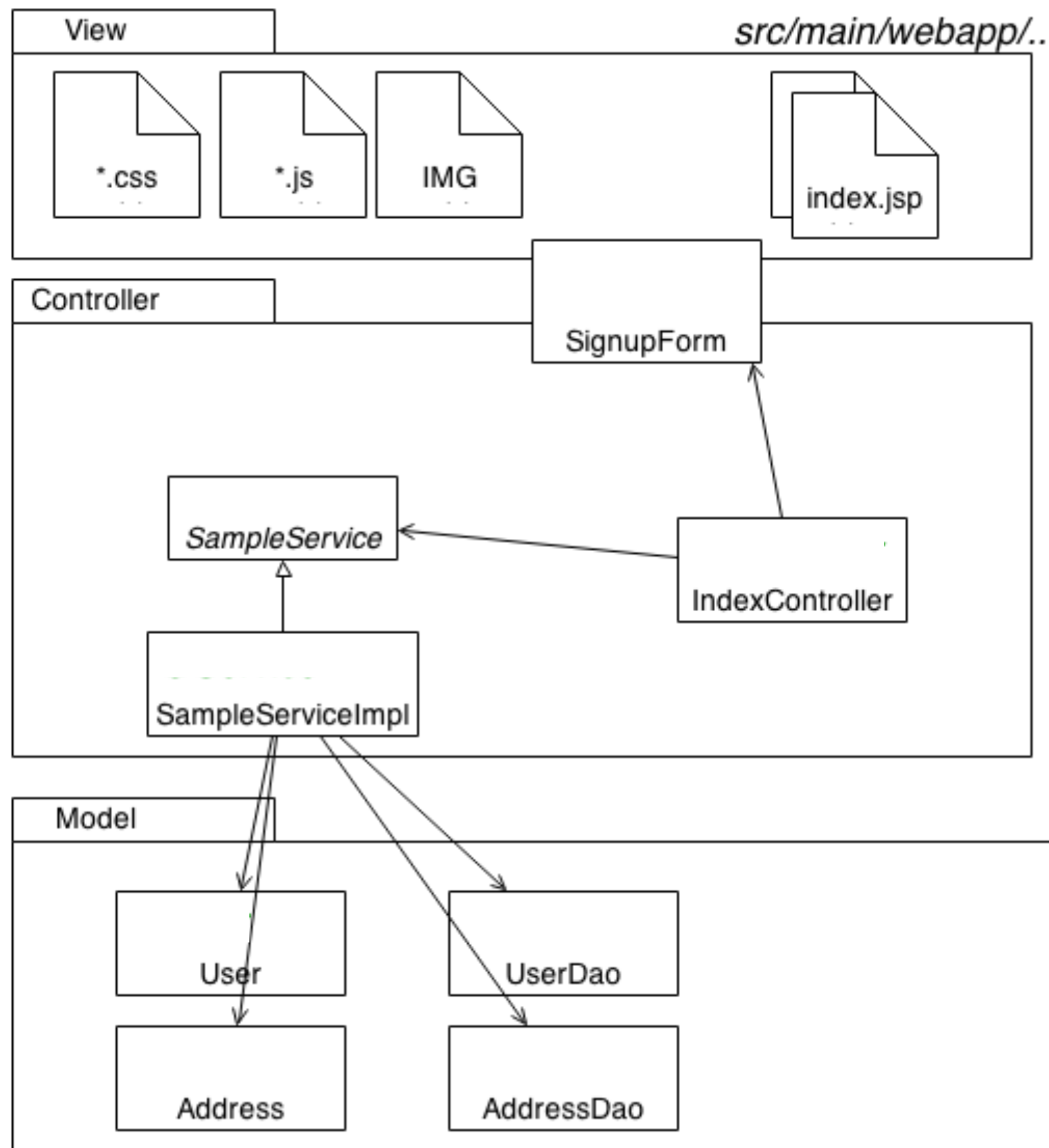
```
<servlet-mapping>  
    <servlet-name>MyPage</..>  
    <url-pattern>/MyPage</..>  
</servlet-mapping>
```



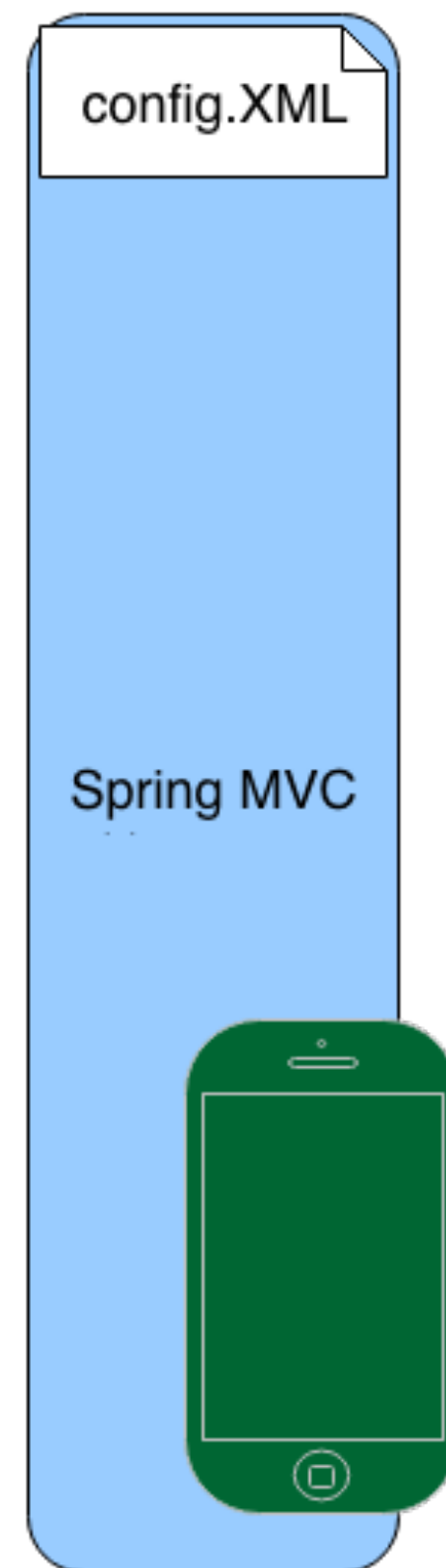
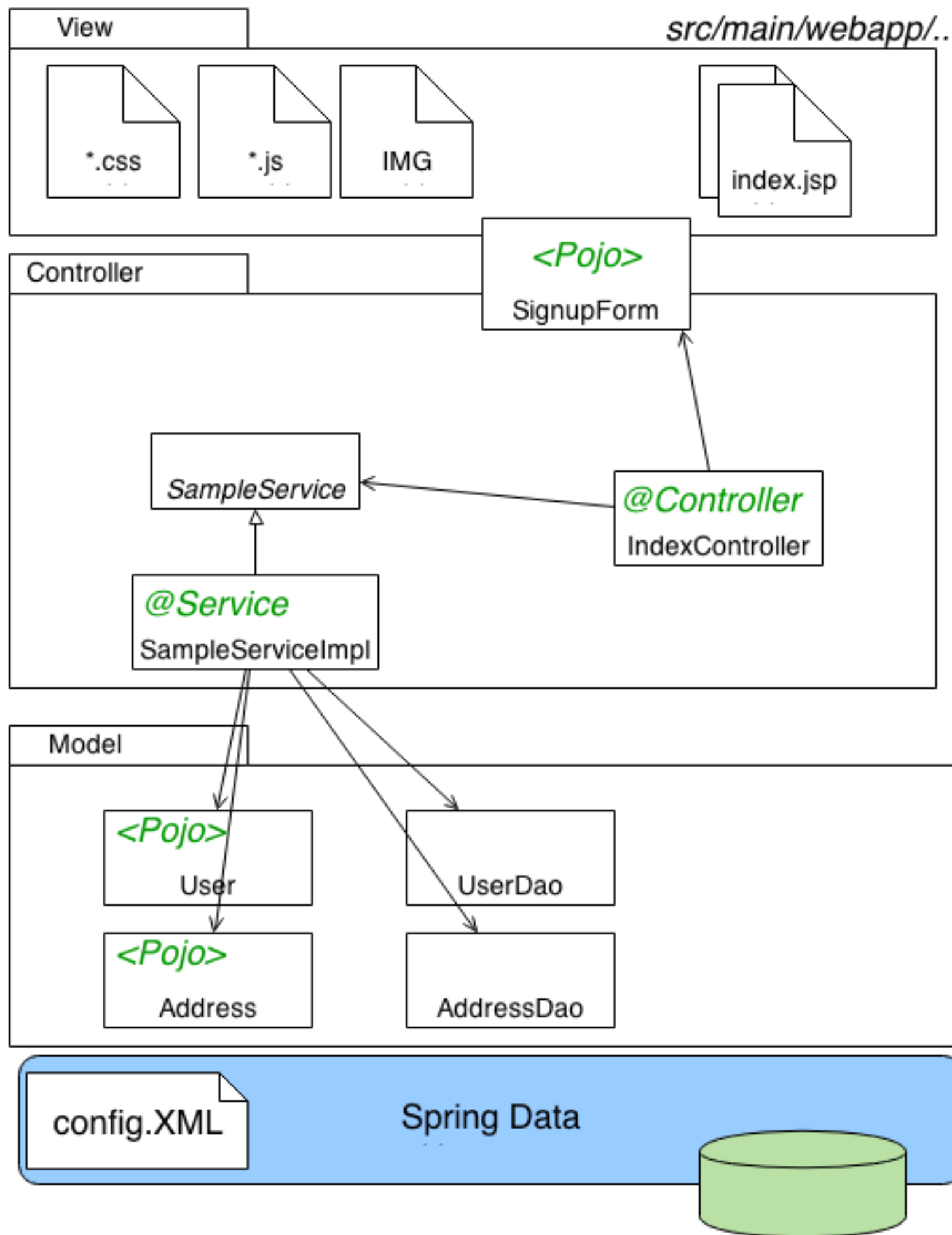
```
public void doPost( HttpServletRequest request,  
                    HttpServletResponse response)  
    throws IOException, ServletException {  
    ....  
}
```

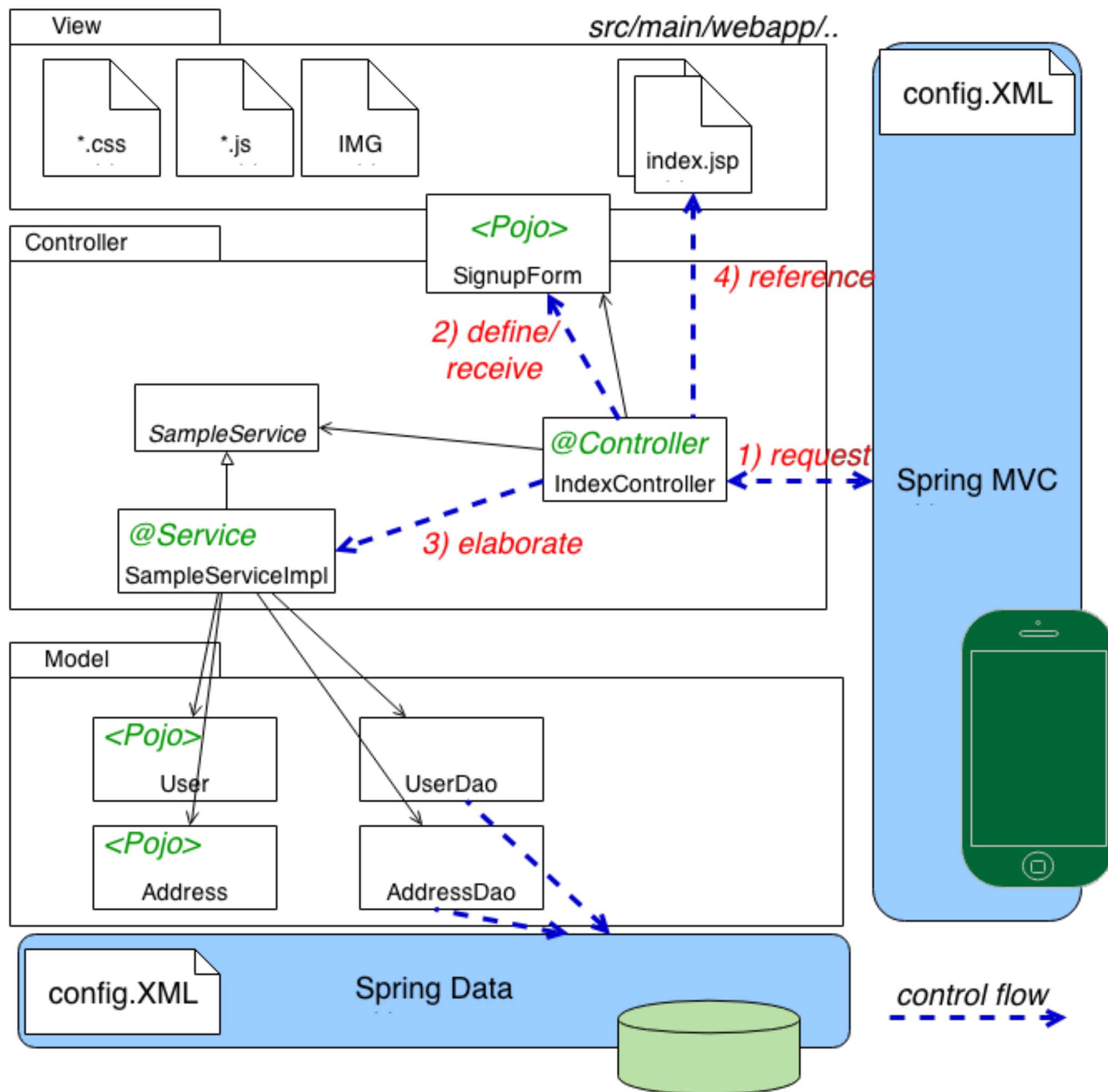


```
insert into myTable(x,y,z) values('foo', 'bar', 'baz')
```



Spring/J2EE





Spring

- **loose-coupling (DI)**
 - example: *@Controller*, *@Autowired*
 - dependencies defined in config / annotation
 - pros: testability, readability, maintainability
- **declarative programming (AOP)**
 - example: *@Transactional*
 - what you want to do vs. how to do it
 - avoid boilerplate code

ESE Skeleton project

- XML

- **pom.xml**: required libraries + build (maven)
- **web.xml**: java webapp deployment descriptor
 - **springMVC.xml**
 - **springData.xml**
 - **springSecurity.xml**

- MVC

- **VIEW**: webapp/pages/**/*.jsp
- **CONTROLLER**: java/org/sample/controller/**/*.java
- **MODEL**: java/org/sample/model/**/*.java

Request

```
@Controller  
public class IndexController {
```

```
@Autowired  
SampleService sampleService;
```

```
@RequestMapping(value = "/", method = RequestMethod.GET)  
public ModelAndView index() {  
    ModelAndView model = new ModelAndView("index");  
    model.addObject("signupForm", new SignupForm());  
    return model;  
}
```

```
}
```

map to request

define required parameters

handle request

return an answer model

java/org/sample/controller/IndexController.java

Request

```
@Controller
public class IndexController {

    @Autowired
    SampleService sampleService;
```

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView index() {
    ModelAndView model = new ModelAndView("index");
    model.addObject("signupForm", new SignupForm());
    return model;
}
```

URI Template Patterns

/owners/fred

/spring-web/spring-web-3.0.5.jar

@RequestMapping(value="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {

@RequestMapping("/spring-web/{symbolicName:[a-z-]}-{version:\\d\\.\\d\\.\\d}{extension:\\.[a-z]}")
public void handle(@PathVariable String version, @PathVariable String extension) {

Consumable Media Types

```
@RequestMapping(value = "/pets", method = RequestMethod.POST, consumes="application/json")
public void addPet(@RequestBody Pet pet, Model model) {
```

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping>

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping-uri-templates>

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping-consumes>

Request

```
@Controller
public class IndexController {


    @Autowired
    SampleService sampleService;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public ModelAndView index() {
        ModelAndView model = new ModelAndView("index");
        model.addObject("signupForm", new SignupForm());
        return model;
    }
}
```

“An `@RequestMapping` handler method can have a very flexible signatures”

Method signature

```
public ModelAndView setupForm(@RequestParam("userId") int petId, ModelAndView model)
public ModelAndView handle(@RequestBody String body, Writer writer) throws IOException
public ModelAndView displayHeaderInfo(@CookieValue("JSESSIONID") String cookie)
public ModelAndView displayHeaderInfo(@RequestHeader("Accept-Encoding") String encoding,
                                       @RequestHeader("Keep-Alive") long keepAlive)
public ModelAndView processSubmit(@Valid Pet pet, BindingResult result)
```

xx.jsp?userId=10 

Handling

```
@RequestMapping(value = "/create", method = RequestMethod.POST)
public ModelAndView create(@Valid SignupForm signupForm, BindingResult
result, RedirectAttributes redirectAttributes) {
    ModelAndView model;
    if (!result.hasErrors()) {
        try {
            sampleService.saveFrom(signupForm);
            model = new ModelAndView("show");
        } catch (InvalidUserException e) {
            model = new ModelAndView("index");
            model.addObject("page_error", e.getMessage());
        }
    } else {
        model = new ModelAndView("index");
    }
    return model;
}
```

pojo validation result

use services to interact with the model

initialize a model

bind user model attributes

return model

java/org/sample/controller/IndexController.java

View

`model.addObject("signupForm", new SignupForm());`
user model attribute (logical name)

target page

```
<form:form method="post" modelAttribute="signupForm" action="create" id="signupForm"
cssClass="form-horizontal" autocomplete="off">
  <fieldset>
    <legend>Enter Your Information</legend>

    <c:set var="emailErrors"><form:errors path="email"/></c:set>
    <div class="control-group"><c:if test="${not empty emailErrors}"> error</c:if></div>
    <label class="control-label" for="field-email">Email</label>

    <div class="controls">
      <form:input path="email" id="field-email" tabindex="1" maxlength="45"
        placeholder="Email"/>
      <form:errors path="email" cssClass="help-inline" element="span"/>
    </div>
  </div>
```

form error

form input field

webapp/pages/index.jsp

Model attribute - Pojo

```
public class SignupForm {  
    private Long id;
```

```
    @NotNull  
    @Pattern(regexp = "[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\\. [a-z0-9!#$%&'*/+=?^_`{|}~-]+)*@(?:  
[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?", message = "Must be  
valid email address")
```

validation rules **



```
    private String email;
```

```
    ..
```

```
    public String getEmail() {  
        return email;  
    }
```

```
    public void setEmail(String email) {  
        this.email = email;  
    }
```

pojo field



getter/setter



```
}
```

** <http://docs.jboss.org/hibernate/validator/4.0.1/reference/en/html/validator-usingvalidator.html#table-builtin-constraints>

java/org/sample/controller/pojos/SignupForm.java

Spring Security

- Use it for authentication and authorization

<http://projects.spring.io/spring-security/>

<http://krams915.blogspot.ch/2010/12/spring-security-3-mvc-using-simple-user.html>

<http://docs.spring.io/spring-security/site/docs/3.2.5.RELEASE/reference/htmlsingle/>

Documentation

17.1. Introduction to Spring Web MVC framework

17.1.1. Features of Spring Web MVC

17.1.2. Pluggability of other MVC implementations

17.2. The DispatcherServlet

17.3. Implementing Controllers

17.3.1. Defining a controller with @Controller

17.3.2. Mapping Requests With @RequestMapping

17.3.3. Defining @RequestMapping handler methods

17.3.4. Asynchronous Request Processing

17.3.5. Testing Controllers

17.4. Handler mappings

17.5. Resolving views

17.5.1. Resolving views with the ViewResolver interface

17.5.2. Chaining ViewResolvers

17.5.3. Redirecting to views

17.5.4. ContentNegotiatingViewResolver

17.6. Using flash attributes

17.7. Building URIs

17.7.1. Building URIs to Controllers and methods

17.7.2. Building URIs to Controllers and methods from views

17.8. Using locales

17.8.1. Obtaining Time Zone Information

17.8.2. AcceptHeaderLocaleResolver

17.8.3. CookieLocaleResolver

17.8.4. SessionLocaleResolver

17.8.5. LocaleChangeInterceptor

17.9. Using themes

17.10. Spring's multipart (file upload) support

17.10.1. Introduction

17.10.2. Using a MultipartResolver with *Commons FileUpload*

17.10.3. Using a MultipartResolver with *Servlet 3.0*

17.10.4. Handling a file upload in a form

17.10.5. Handling a file upload request from programmatic clients

17.11. Handling exceptions

17.11.1. HandlerExceptionResolver

17.11.2. @ExceptionHandler

17.11.3. Handling Standard Spring MVC Exceptions

17.11.4. Annotating Business Exceptions With @ResponseStatus

17.11.5. Customizing the Default Servlet Container Error Page

17.12. Web Security

17.13. Convention over configuration support

17.14. ETag support

17.15. Code-based Servlet container initialization

17.16. Configuring Spring MVC

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>