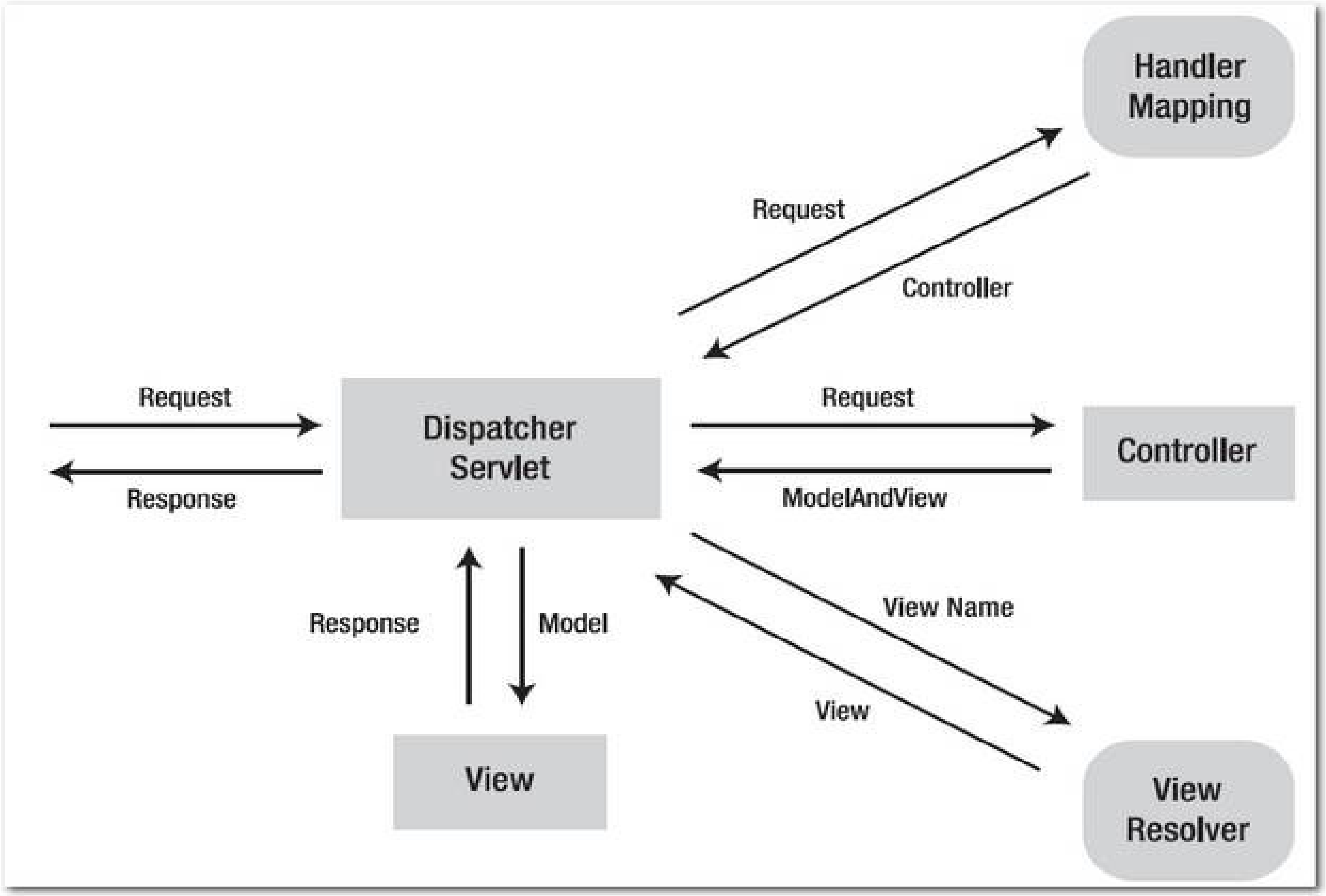
Spring Framework

Understanding Model View Controller (MVC)

Team Emertxe

Spring MVC

Spring Architecture



Exploring Spring MVC Framework

- The Spring Web MVC framework is built on generic Servlet known as a DispatcherServlet class(Front Controller).
- The DispatcherServlet class sends the request to the handlers with configurable handler mappings, theme resolution, locale and view resolution along with file uploading.

Exploring Spring's Web MVC Framework

- The handleRequest(request, response) method is given by the default handler called Controller interface.
- The application controller implementation classes of the controller interface are as follows:

  – AbstractController

  – AbstractCommandController

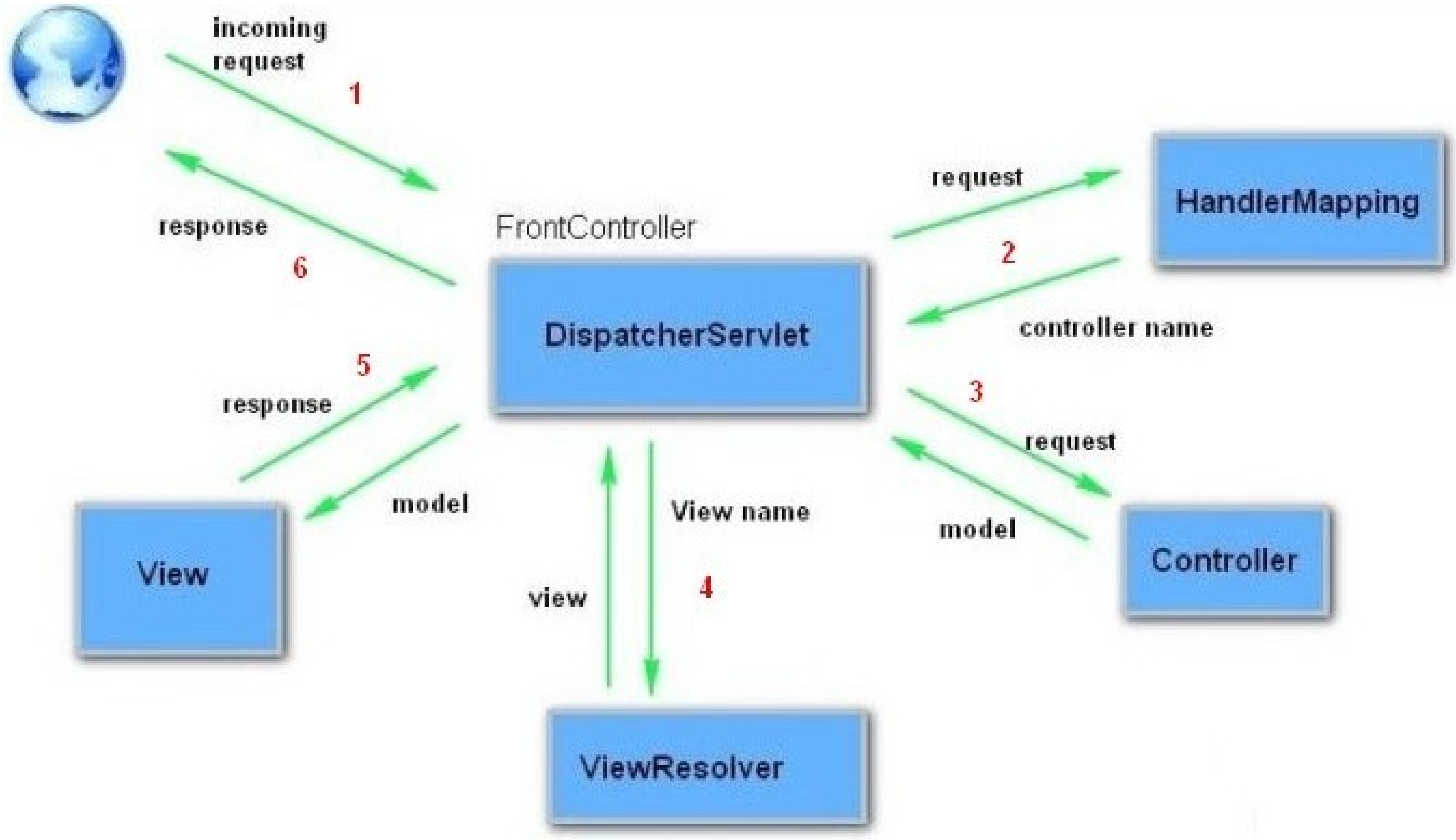
  – SimpleFormController.

Spring MVC Features

- Powerful configuration of both framework and application classes.
- Separation of roles.
- Flexibility in choosing subclasses.
- Model transfer flexibility.
- No need of duplication of code.
- Specific validation and binding.
- Specific local and theme resolution.
- Facility of JSP form tag library.

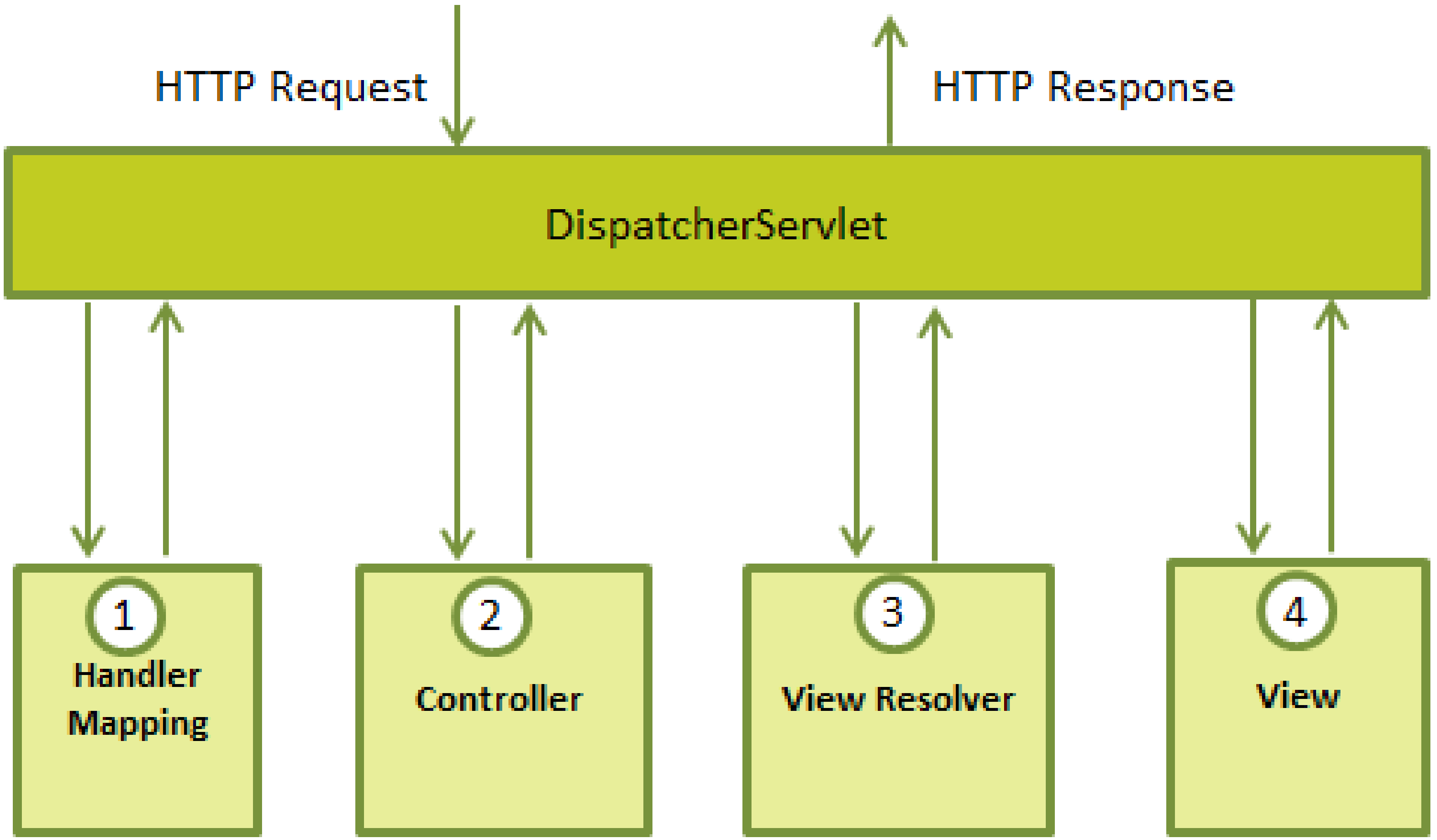
Flow of Spring MVC

Flow of Spring MVC



DispatcherServlet

DispatcherServlet



DispatcherServlet

- The DispatcherServlet class is important part of spring Web MVC framework.
- It is used for dispatching the request to application controllers.
- The DispatcherServlet class is configured in web.xml file of a web application.
- Map the request using Uniform Resource Locator (URL) mapping in the same web.xml file to handle any request

HadlerMapping/Controller /ViewResolver

- Handler mapping : Manages the execution of controllers, provided they match the specified criteria.
- Controller : It handles the client's request.
- View resolver : Resolves view names to view used by the DispatcherServlet.

Creating Spring MVC Application

Creating Spring MVC Application

- Create the request page (optional)
- Create the controller class
- Provide the entry of controller in the web.xml file
- Define the bean in the XML file
- Display the message in the JSP page
- Load the spring core and MVC jar files
- Start server and deploy the project

Controller Class

```
@Controller
public class HelloWorldController
{
@RequestMapping("/samplepage")
public ModelAndView helloSpring()
{
String message = "Welcome to Spring MVC";
return new ModelAndView("samplepage", "message",
message);
}
}
```

Controller Annotation

- The @Controller annotation defines the class as a Spring MVC controller.
- The @RequestMapping annotation is used to map URLs like '/hello' onto an entire class or a particular handler method.
- @RequestMapping(method = RequestMethod.GET) is used to declare the printHello() method as the controller's default service method to handle HTTP GET request.

web.xml File

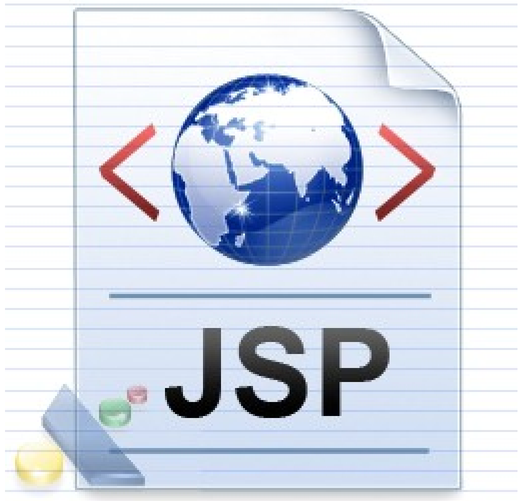
```
<servlet>
<servlet-name>spring</servlet-name>
<servlet-class>
 org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>spring</servlet-name>
<url-pattern>*.html</url-pattern>
</servlet-mapping>
```

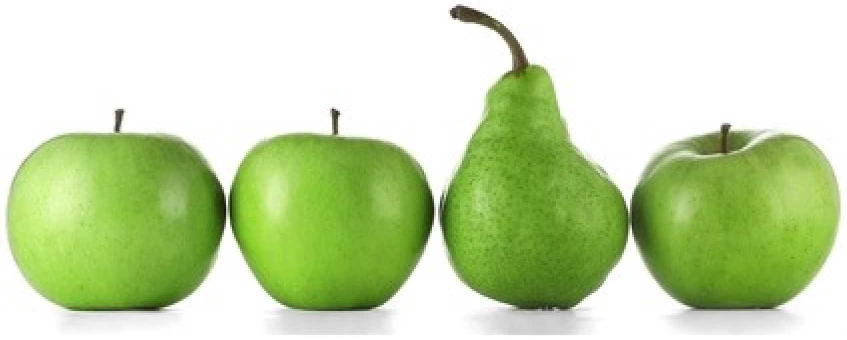
spring-servlet.xml

```
<context:component-scan basepackage="com.sample"/>

<bean class="org.springframework.web.servlet.view.

InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" />

</bean>
```

JSP Page

- Message is: ${message}//sample.jsp
- This is the simple JSP page, displaying the message returned by the Controller.
- It must be located inside the WEB-INF/jsp directory .
- Finally load the jar files and run.





Spring Exception Handling

Spring Exception

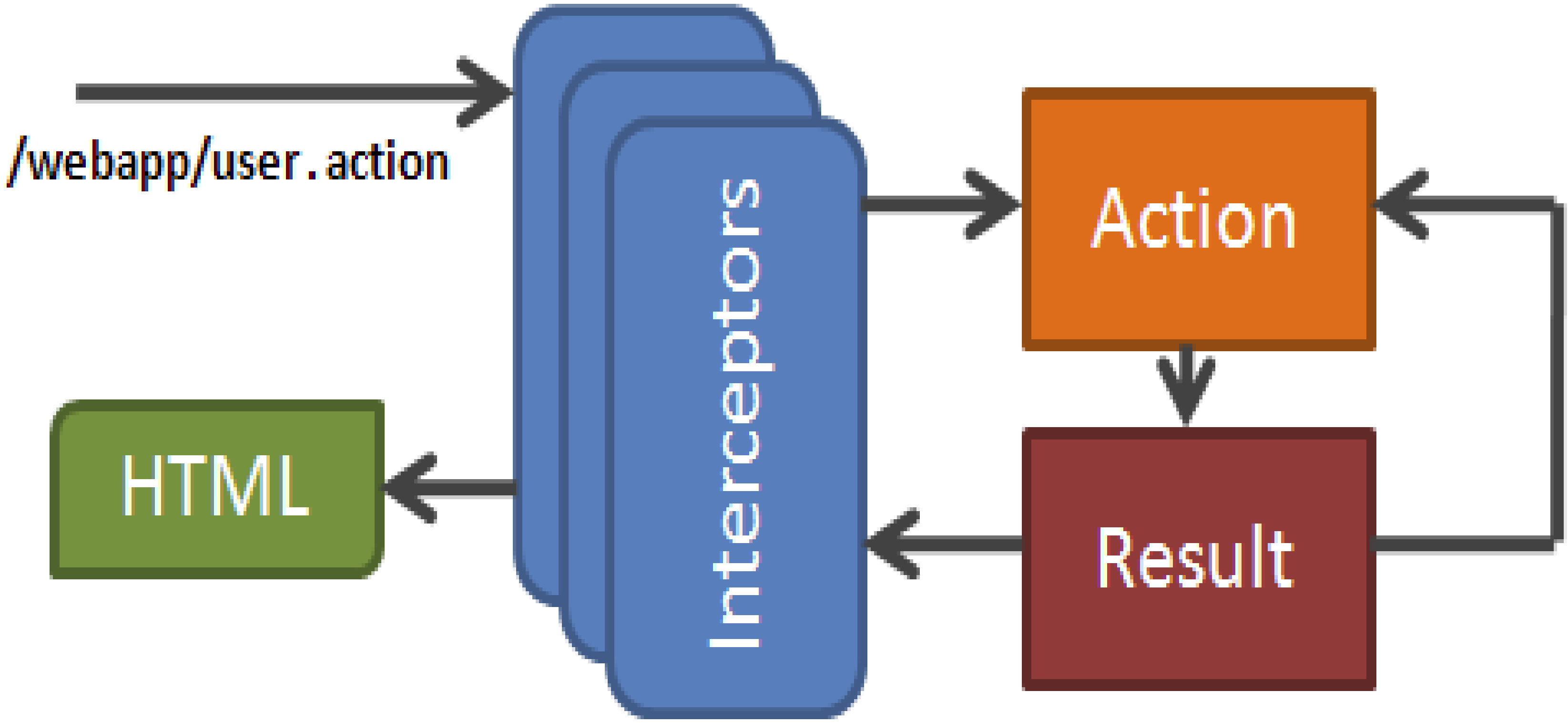
- @ExceptionHandler only handles exception getting raised from the controller where it is defined.
- It will not handle exceptions getting raised from other controllers. @ControllerAdvice annotation solves this problem.
- @ControllerAdvice annotation is used to define @ExceptionHandler, @InitBinder, and @ModelAttribute methods that apply to all @RequestMapping methods.

Spring Exception

```
import  org.springframework.web.bind.annotation.ControllerAdvice;
@ControllerAdvice
public class ExceptionControllerAdvice
{
@ExceptionHandler(Exception.class)
public String exception(Exception e)
{ return "error";}
}
```

Spring Interceptor

Spring Interceptor



Spring Interceptor

- Spring MVC provides a powerful mechanism to intercept an http request
- Each interceptor you define must implement org.springframework.web.servlet.HandlerInterceptor interface.
- HandlerInterceptor – an interface, which must be implemented by the Spring interceptor classes, has the following three methods.
  - preHandle(...) – called just before the controller
  - postHandle(...) – called immediately after the controller
  - afterCompletion(...) – called just before sending response to view

Spring Interceptor

- HandlerInterceptorAdaptor – an implementation class of HandlerInterceptor interface provided by Spring as a convenient class. By extending this we can override only the necessary methods out of the three.

Spring Validation

Spring Validation

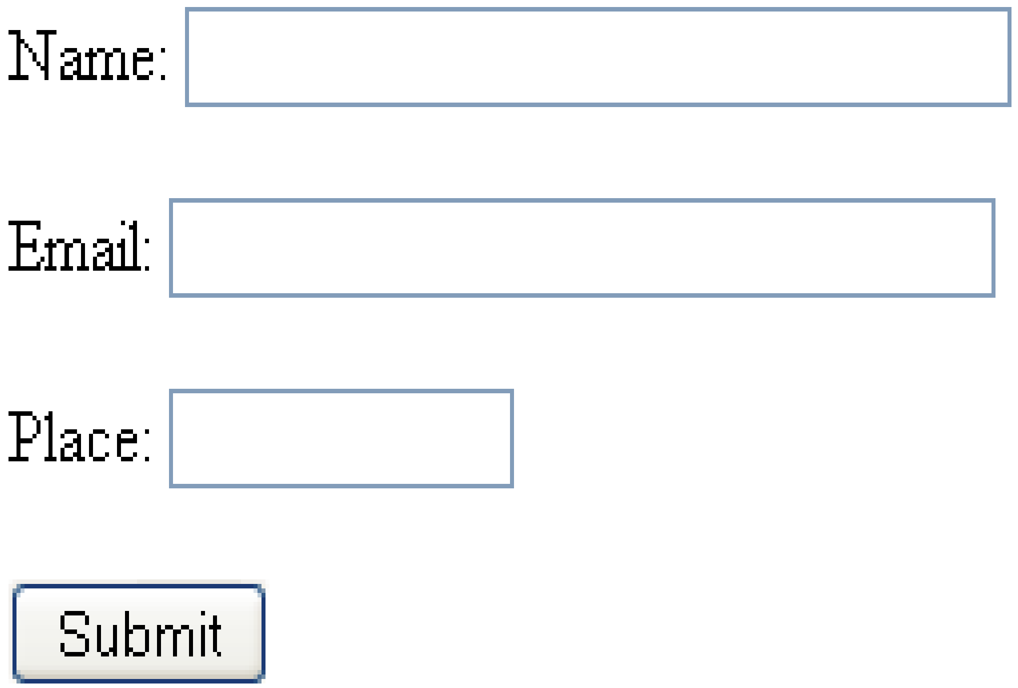
- Spring provides a simplified set of APIs and supporting classes for validating domain objects.
- Spring features a Validator interface that you can use to validate objects. The Validator interface works using an Errors object so that while validating, validators can report validation failures to the Errors object.

Spring Validation

```
public class UserLoginValidator implements Validator
{
public boolean supports(Class clazz)
{
//............................
}
public void validate(Object target, Errors errors)
{
//........................................
}
}
```

Spring's Form Tag Library

Spring Form Tag Lib



Spring Form Tag Lib

- Spring MVC provides a JSP tag library for making it easier to bind form elements to Model data.
- Spring Framework also provides you with some tags for evaluating errors, setting themes and outputting internationalized messages.
- Syntax:

```
<%@taglib uri="http://www.springframework.org/tags/form"
prefix="form">
```

Spring Form Tag

- Tag <form:form> tag, replaces html tag form.
- It takes the addition attribute <modelAttribute> which is the name of the form object whose properties are used to populate the form.
- It may specify any attribute added to the Model object by the controller handler that selects the form view.
- <form:form method="POST" action="url-path" modelAttribute="a">
- ...
- </form:form>

Text Input and Output Tags

- The tag <form:input> is used for text input, and replaces the HTML tag input.
- It introduces the extra attribute path that specifies the field or field path to be accessed.
- It is similar to the attribute value in HTML.
- `<form:input path="prop1" />`

EMERTXE

Multi-Option Selection Tags

- Multiple option selection is declared with tag <form:select>. This tag is usually rendered as a combo-box.

  Example:

  ```
  <form:select path="a">
  <form:option label="5" value="5"  />
  <form:option label="10" value="10"  />
  </form:select>
  ```

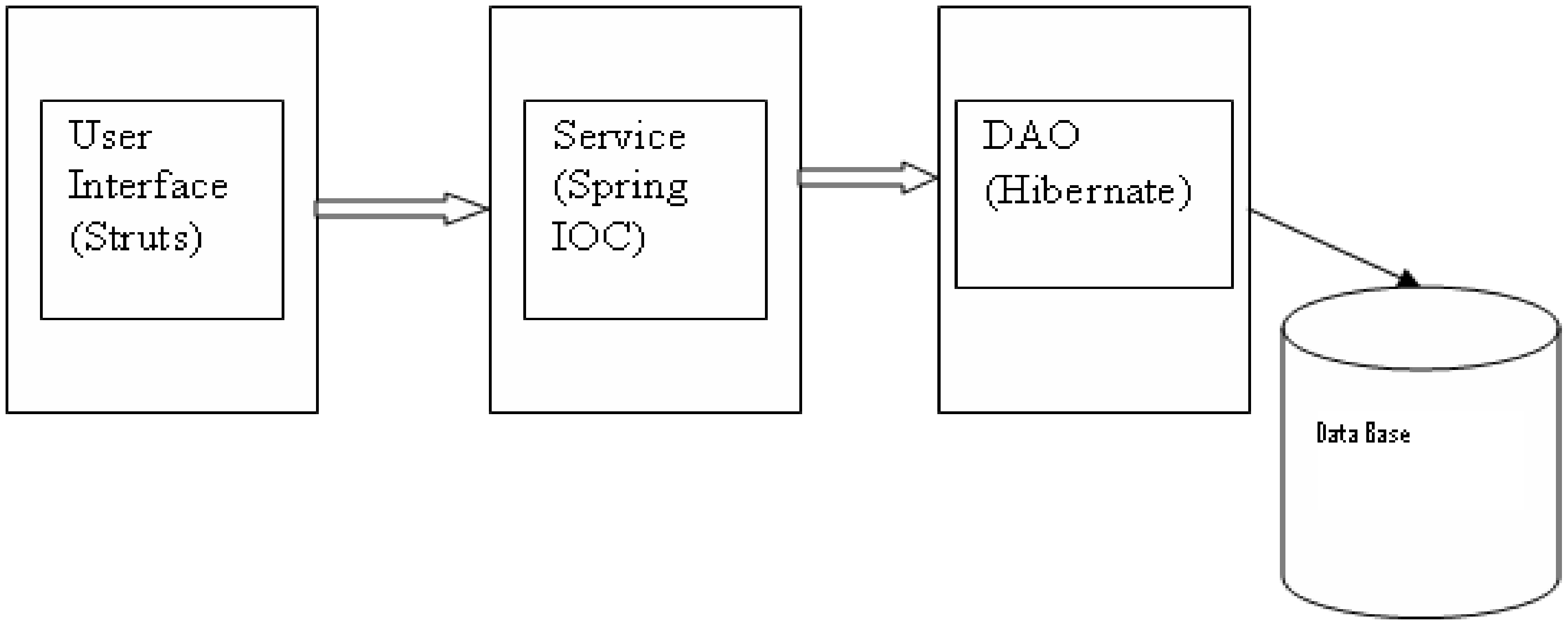
Integrating Spring And Hibernate

Spring with Hibernate

- When we integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file. We can provide all the information in the applicationContext.xml file.
- It saves lots of code.
- We don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc.

```
Student s1=new Student(1,"park",25000);

hibernateTemplate.save(s1);
```

Struts, Spring and Hibernate



Questions

- What are Features of spring?
- What is Controller?
- Explain DispatcherServlet?

ΣMERTXE

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe

https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

Thank You