Вопросы и ответы на собеседование по Spring

(16-30)

16). Как выглядит типичная реализция метода используя Spring?

Для типичного Spring-приложения нам необходимы следующие файлы:

- Интерфейс, описывающий функционал приложения
- Реализация интерфейса, содержащая свойства, сэттеры-гэттеры, функции и т.п.
- Конфигурационный XML-файл Spring'a.
- Клиентское приложение, которое использует функцию.

17). Что такое связывание в Spring и расскажите об аннотации @Autowired?

Процесс внедрения зависимостей в бины при инициализации называется Spring Bean Wiring.
Считается хорошей практикой задавать явные связи между зависимостями, но в Spring предусмотрен дополнительный механизм связывания **@Autowired**. Аннотация может использоваться над **полем** или **методом** для связывания по типу. Чтобы аннотация заработала, необходимо указать небольшие настройки в конфигурационном файле спринг с помощью элемента **context:annotation-config.**

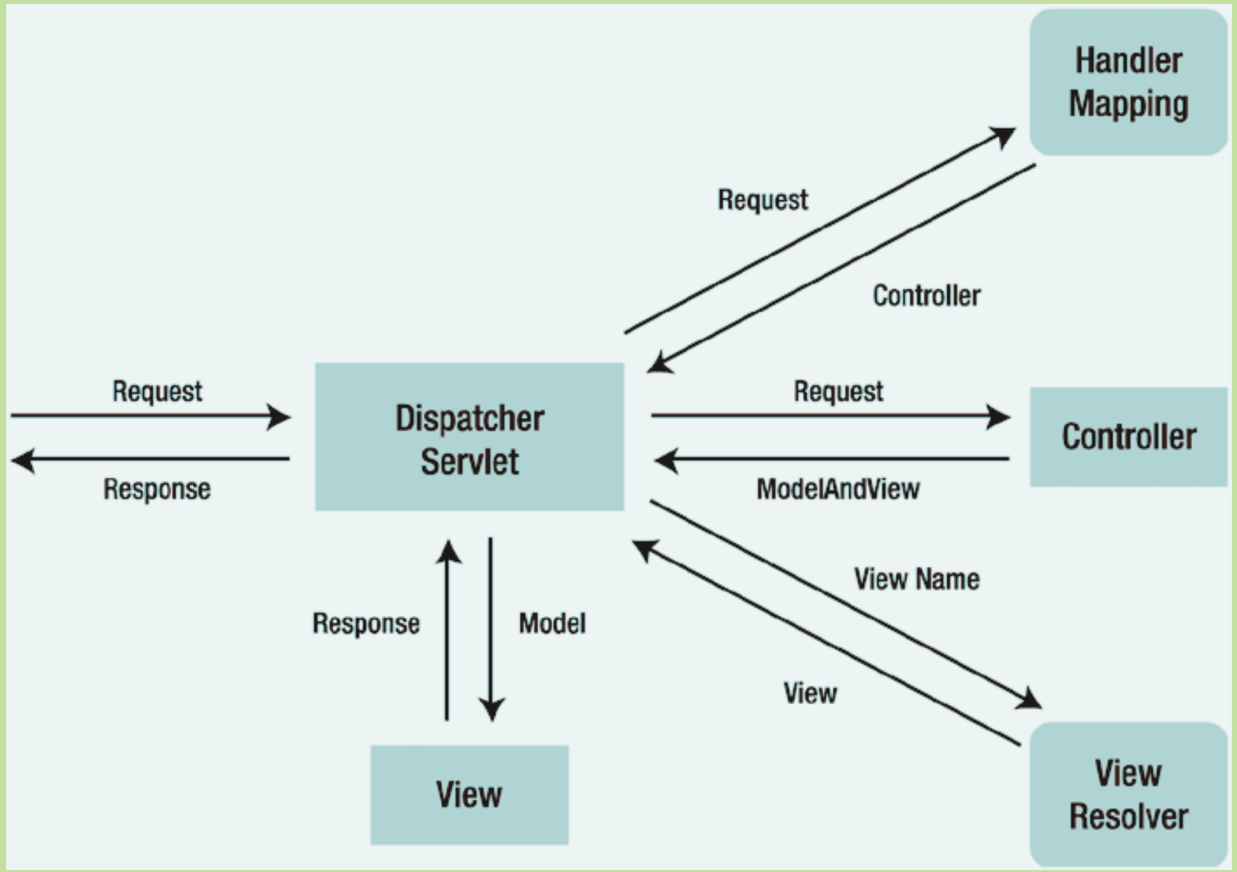
18). Какие различные типы автоматического связывания в Spring?

Существует четыре вида связывания в спринг:

- autowire byName,
- autowire byType,
- autowire by constructor,
- autowiring by @Autowired and @Qualifier annotations

19). Что такое контроллер в Spring MVC?

Ключевым интерфейсом в Spring MVC является **Controller**. Контроллер обрабатывает запросы к действиям, осуществляемые пользователями в пользовательском интерфейсе, взаимодействуя с уровнем обслуживания, обновляя модель и направляя пользователей на соответствующие представления в зависимости от результатов выполнения. Controller — **управление, связь между моделью и видом.**



Проще говоря, все запросы, обрабатываемые **DispatcherServlet** , направляются к классам, аннотированным **@Controller** . Каждый класс контроллера сопоставляет один или несколько запросов с методами, которые обрабатывают и выполняют запросы с предоставленными входными данными.

20). Какая разница между аннотациями @Component, @Repository и @Service в Spring?

@Component используется для указания класса в качестве компонента спринг. При использовании поиска аннотаций, такой класс будет сконфигурирован как spring bean.

@Controller специальный тип класса, применяемый в MVC приложениях. Обрабатывает запросы и часто используется с аннотацией @RequestMapping.

@Repository указывает, что класс используется для работы с поиском, получением и хранением данных. Аннотация может использоваться для реализации шаблона DAO.

@Service указывает, что класс является сервисом для реализации бизнес логики (на самом деле не отличается от Component, но просто помогает разработчику указать смысловую нагрузку класса).

Для указания контейнеру на класс-бин можно использовать любую из этих аннотаций. Но различные имена позволяют различать назначение того или иного класса.

21). Расскажите, что вы знаете о DispatcherServlet и ContextLoaderListener

DispatcherServlet — сервлет диспетчера. Этот сервлет анализирует запросы и направляет их соответствующему контроллеру для обработки. В Spring MVC класс DispatcherServlet является центральным **сервлетом**, который получает запросы и направляет их соответствующим **контроллерам**. В приложении Spring MVC может существовать произвольное количество экземпляров DispatcherServlet, предназначенных для разных целей (например, для обработки запросов пользовательского интерфейса, запросов веб-служб REST и т.д.). Каждый экземпляр DispatcherServlet имеет собственную конфигурацию **WebApplicationContext**, которая определяет характеристики уровня сервлета, такие как контроллеры, поддерживающие сервлет, отображение обработчиков, распознавание представлений, интернационализация, оформление темами, проверка достоверности, преобразование типов и форматирование и т.п.
ContextLoaderListener — слушатель при старте и завершении корневого класса Spring WebApplicationContext. Основным назначением является связывание жизненного цикла ApplicationContext и ServletContext, а так же автоматического создания ApplicationContext. Можно использовать этот класс для доступа к бинам из различных контекстов спринг. Настраивается в **web.xml**:

```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<listener>
   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

22). Что такое ViewResolver в Spring?

ViewResolver — распознаватель представлений. Интерфейс ViewResolver в Spring MVC (из пакета **org.springframework.web.servlet**) поддерживает распознавание представлений на основе логического имени, возвращаемого контроллером. Для поддержки различных механизмов распознавания представлений предусмотрено множество классов реализации. Например, класс **UrlBasedViewResolver** поддерживает прямое преобразование логических имен в URL.
Класс **ContentNegotiatingViewResolver** поддерживает динамическое распознавание представлений в зависимости от типа медиа, поддерживаемого клиентом (XML, PDF, JSON и т.д.). Существует также несколько реализаций для интеграции с различными технологиями представлений, такими как FreeMarker (**FreeMarkerViewResolver**), Velocity (**VelocityViewResolver**) и JasperReports (**JasperReportsViewResolver**).

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources
        in the /WEB-INF/views directory -->
    <bean
          class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
```

InternalResourceViewResolver — реализация **ViewResolver**, которая позволяет находить представления, которые возвращает контроллер для последующего перехода к нему. Ищет по заданному пути, префиксу, суффиксу и имени.

23). Что такое MultipartResolver и когда его использовать?

Интерфейс **MultipartResolver** используется для загрузки файлов. Существуют две реализации: **CommonsMultipartResolver** и **StandardServletMultipartResolver**, которые позволяют фреймворку загружать файлы. По умолчанию этот интерфейс не включается в приложении и необходимо указывать его в файле конфигурации. После настройки любой запрос о загрузке будет отправляться этому интерфейсу.

```
<beans:bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

      <!-- setting maximum upload size -->
      <beans:property name="maxUploadSize" value="100000" />

 </beans:bean>
```

24). Как обрабатывать исключения в Spring MVC Framework?

В Spring MVC интерфейс **HandlerExceptionResolver** (из пакета **org.springframework.web.servlet**) предназначен для работы с непредвиденными исключениями, возникающими во время выполнения обработчиков.

По умолчанию **DispatcherServlet** регистрирует класс **DefaultHandlerExceptionResolver** (из пакета **org.springframework.web.servlet.mvc.support**).
Этот распознаватель обрабатывает определенные стандартные исключения Spring MVC, устанавливая специальный код состояния ответа. Можно также реализовать собственный обработчик исключений, аннотировав метод контроллера с помощью аннотации **@ExceptionHandler** и передав ей в качестве атрибута тип исключения. В общем случае обработку исключений можно описать таким образом:

Controller Based – указать методы для обработки исключения в классе контроллера. Для этого нужно пометить такие методы аннотацией **@ExceptionHandler**.

Global Exception Handler – для обработки глобальных исключений спринг предоставляет аннотацию **@ControllerAdvice**.

HandlerExceptionResolver implementation – Spring Framework предоставляет интерфейс **HandlerExceptionResolver**, который позволяет задать глобального обработчика исключений. Реализацию этого интерфейса можно использовать для создания собственных глобальных обработчиков исключений в приложении.

25). Можем ли мы иметь несколько файлов конфигурации Spring?

С помощью указания contextConfigLocation можно задать несколько файлов конфигурации Spring. Параметры указываются через **запятую** или **пробел**:

```
<servlet>
   <servlet-name>appServlet</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   <init-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>/WEB-INF/spring/appServlet/servlet-context.xml,/WEB-INF/spring/appServlet/servlet-jdbc.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

Поддерживается возможность указания нескольких корневых файлов конфигурации Spring:

```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>/WEB-INF/spring/root-context.xml /WEB-INF/spring/root-security.xml</param-value>
</context-param>
```

Файл конфигурации можно импортировать:

```
<beans:import resource="spring-jdbc.xml"/>
```

26). Какие минимальные настройки нужны, чтобы создать приложение Spring MVC?

Для создания простого Spring MVC приложения необходимо пройти следующие шаги:

- Добавить зависимости spring-context и spring-webmvc в проект.
- Указать DispatcherServlet в web.xml для обработки запросов внутри приложения.
- Задать определение spring bean (аннотацией или в xml). Добавить определение view resolver для представлений.
- Настроить класс контроллер для обработки клиентских запросов.

27). Как бы вы связали Spring MVC Framework и архитектуру MVC?

- Моделью (**Model**) выступает любой Java bean в Spring. Внутри класса могут быть заданы различные атрибуты и свойства для использования в представлении.
- Преставление (**View**) — JSP страница, HTML файл и т.п. служат для отображения необходимой информации пользователю. Представление передает обработку запросов к диспетчеру сервлетов (контроллеру).
- DispatcherServlet (**Controller**) — это главный контроллер в приложении Spring MVC, который обрабатывает все входящие запросы и передает их для обработки в различные методы в контроллеры.

28). Как добиться локализации в приложениях Spring MVC?

Spring MVC предоставляет очень простую и удобную возможность локализации приложения. Для этого необходимо сделать следующее:

- Создать файл resource bundle, в котором будут заданы различные варианты локализированной информации.
- Определить messageSource в конфигурации Spring используя классы ResourceBundleMessageSource или ReloadableResourceBundleMessageSource.
- Определить localeResolver класса CookieLocaleResolver для включения возможности переключения локали.
- С помощью элемента spring:message DispatcherServlet будет определять в каком месте необходимо подставлять локализированное сообщение в ответе.

```
<beans:bean id="messageSource"
   class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
   <beans:property name="basename" value="classpath:messages" />
   <beans:property name="defaultEncoding" value="UTF-8" />
</beans:bean>

<beans:bean id="localeResolver"
   class="org.springframework.web.servlet.i18n.CookieLocaleResolver">
   <beans:property name="defaultLocale" value="en" />
   <beans:property name="cookieName" value="myAppLocaleCookie"></beans:property>
   <beans:property name="cookieMaxAge" value="3600"></beans:property>
</beans:bean>

<interceptors>
   <beans:bean
      class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
      <beans:property name="paramName" value="locale" />
   </beans:bean>
</interceptors>
```

29). Как мы можем использовать Spring для создания веб-службы RESTful, возвращающей JSON?

Spring Framework позволяет создавать Restful веб сервисы и возвращать данные в формате JSON. Spring обеспечивает интеграцию с Jackson JSON API для возможности отправки JSON ответов в restful web сервисе. Для отправки ответа в формате JSON из Spring MVC приложения необходимо произвести следующие настройки:

- Добавить зависимости Jackson JSON. С помощью maven это делается так:

```
<!-- Jackson -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>${jackson.databind-version}</version>
</dependency>
```

29). Продолжение

- Настроить бин RequestMappingHandlerAdapter в файле конфигурации Spring и задать свойство messageConverters на использование бина MappingJackson2HttpMessageConverter.

```
<!-- Configure to plugin JSON as request and response in method handler -->
<beans:bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
    <beans:property name="messageConverters">
        <beans:list>
            <beans:ref bean="jsonMessageConverter"/>
        </beans:list>
    </beans:property>
</beans:bean>

<!-- Configure bean to convert JSON to POJO and vice versa -->
<beans:bean id="jsonMessageConverter"
class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">
</beans:bean>
```

- В контроллере указать с помощью аннотации @ResponseBody возвращение Object:

```
@RequestMapping(value = EmpRestURIConstants.GET_EMP, method = RequestMethod.GET)
public @ResponseBody Employee getEmployee(@PathVariable("id") int empId) {
    logger.info("Start getEmployee. ID="+empId);

    return empData.get(empId);
}
```

30). Можем ли мы послать объект как ответ метода обработчика контроллера?

Да, это возможно. Для этого используется аннотация **@ResponseBody**. Так можно отправлять ответы в виде JSON, XML в restful веб сервисах.