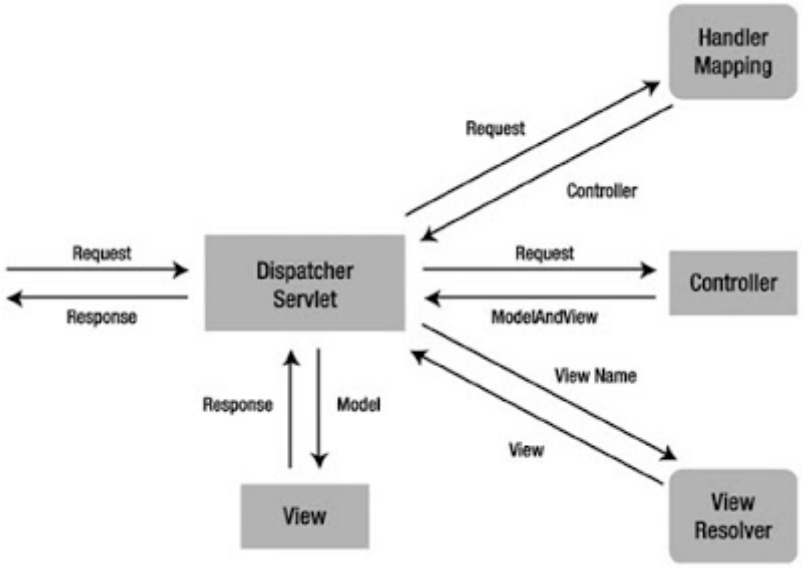
INTERVIEW JAVA DEVELOPER (ADVANCED) SPRING FRAMEWORK

Ответы на вопросы на собеседование Spring Framework (часть 3).

Vasyl K 8:09:00 Добавить комментарий

- **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.



Основным контроллером в Spring MVC является org.springframework.web.servlet.DispatcherServlet. Задается аннотацией @Controller и часто используется с аннотацией @RequestMapping, которая указывает какие запросы будут обрабатываться этим контроллером.

- **КАКАЯ РАЗНИЦА МЕЖДУ АННОТАЦИЯМИ @COMPONENT, @REPOSITORY И @SERVICE В SPRING?**

@Component – используется для указания класса в качестве компонента spring. 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.

- **РАССКАЖИТЕ, ЧТО ВЫ ЗНАЕТЕ О DISPATCHERSERVLET И CONTEXTLOADERLISTENER.**

DispatcherServlet – сервлет диспатчер. 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>
```

- **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">
</property></property></bean>
```

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

- **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:property></beans:bean>
```

- **КАК ЗАГРУЗИТЬ ФАЙЛ В SPRING MVC?**

Внутри спринг предусмотрен интерфейс MultipartResolver для обеспечения загрузки файлов. Фактически нужно настроить файл конфигурации для указания обработчика загрузки файлов, а затем задать необходимый метод в контроллере spring.

- **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 контроллере. Для этого нужно пометить такие методы аннотацией @ExceptionHandler.
Global Exception Handler – для обработки глобальных исключений spring предоставляет аннотацию @ControllerAdvice.
HandlerExceptionResolver implementation – общие исключений большая часть времени обслуживают статические страницы. Spring Framework предоставляет интерфейс HandlerExceptionResolver, который позволяет задать глобального обработчика исключений. Реализацию этого интерфейса можно использовать для создания собственных глобальных обработчиков исключений в приложении.

- **КАКОВЫ МИНИМАЛЬНЫЕ НАСТРОЙКИ, 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.

- **КАК БЫ ВЫ СВЯЗАЛИ SPRING MVC FRAMEWORK И АРХИТЕКТУРУ MVC?**

Модель (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.

- **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 ResourceBundleMessageSource.
- Определить localceResolver 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:property></beans:property></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:property></beans:bean>

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

- **КАК МЫ МОЖЕМ ИСПОЛЬЗОВАТЬ SPRING ДЛЯ СОЗДАНИЯ ВЕБ-СЛУЖБЫ RESTFUL, ВОЗВРАЩАЮЩЕЙ JSON?**

Spring Framework позволяет создавать Resful веб сервисы и возвращать данные в формате JSON. Spring обеспечивает интеграцию с Jackson JSON API для возможности отправки JSON ответов в resful 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>
```

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:ref></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:

```java
@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);
}
```

- **КАК ПРОВЕРИТЬ (ВАЛИДИРОВАТЬ) ДАННЫЕ ФОРМЫ В SPRING WEB MVC FRAMEWORK?**

Spring поддерживает аннотации валидации из JSR-303, а так же возможность создания своих реализаций классов валидаторов. Пример использования аннотаций:

```java
@Size(min=2, max=30)
private String name;

@NotEmpty @Email
private String email;

@NotNull @Min(18) @Max(100)
private Integer age;

@NotNull
private Gender gender;

@DateTimeFormat(pattern="MM/dd/yyyy")
@NotNull @Past
private Date birthday;
```

- **ЧТО ВЫ ЗНАЕТЕ SPRING MVC INTERCEPTOR И КАК ОН ИСПОЛЬЗУЕТСЯ?**

Перехватчики в Spring (Spring Interceptor) являются аналогом Servlet Filter и позволяют перехватывать запросы клиента и обрабатывать их. Перехватить запрос клиента можно в трех местах: preHandle, postHandle и afterCompletion.

- preHandle – метод используется для обработки запросов, которые еще не были переданы в метода обработчик контроллера. Должен вернуть true для передачи следующему перехватчику или в handler method. False укажет на обработку запроса самим обработчиком и отсутствию необходимости передавать его дальше. Метод имеет возможность выкидывать исключения и пересылать ошибки к представлению.
- postHandle – вызывается после handler method, но до обработки DispatcherServlet для передачи представлению. Может использоваться для добавления параметров в объект ModelAndView.
- afterCompletion – вызывается после отрисовки представления.

Для создания обработчика необходимо расширить абстрактный класс HandlerInterceptorAdapter или реализовать интерфейс HandlerInterceptor. Так же нужно указать перехватчики в конфигурационном файле Spring.

```xml
<!-- Configuring interceptors based on URI -->
<interceptors>
    <interceptor>
        <mapping path="/home">
        <beans:bean class="jsehelper.spring.RequestProcessingTimeInterceptor"></beans:bean>
    </mapping></interceptor>
</interceptors>
```

- **РАССКАЖИТЕ О SPRING SECURITY.**

Проект Spring Security предоставляет широкие возможности для защиты приложения. Кроме стандартных настроек для аутентификации, авторизации и распределения ролей и маппинга доступных страниц, ссылок и т.п., предоставляет защиту от различных вариантов атак (например CSRF). Имеет множество различных настроек, но остается легким в использовании.