JAVA + SPRING MVC

שיעור 04: ViewResolvers

1. חזרה על HandlerMapping

באמצעות handler mapping ניתן למפות בקשות אינטרנט נכנסות למטפלים מתאימים. יש כמה handler mapping שבהם אתה יכול להשתמש מחוץ לקופסה, למשל, SimpleUrlHandlerMapping או BeanNameUrlHandlerMapping, אבל תחילה נבחן את הרעיון הכללי של HandlerMapping.

הפונקציונליות ש- HandlerMapping הבסיסית מספקת היא אספקת HandlerExecutionChain, אשר חייב להכיל את המטפל שתואם את הבקשה הנכנסת, ויכול להכיל גם רשימה של מיירטי מטפל המוחלים על הבקשה. כשמגיעה בקשה, ה-DispatcherServlet ימסור אותה למיפוי המטפל כדי לאפשר לו לבדוק את הבקשה ולהמציא HandlerExecutionChain מתאים. אז ה-DispatcherServlet יבצע את המטפל והמיירטים בשרשרת (אם יש).

הרעיון של מיפויים ניתנים להגדרה של מטפל שיכולים להכיל מיירטים באופן אופציונלי (שבוצע לפני או אחרי ביצוע המטפל בפועל, או שניהם) הוא חזק ביותר. ניתן לבנות הרבה פונקציונליות תומכת ב- HandlerMappings מותאמים אישית. חשבו על מיפוי מטפל מותאם אישית שבוחר מטפל לא רק על סמך כתובת ה-URL של הבקשה שנכנסה, אלא גם על מצב ספציפי של ההפעלה המשויכת לבקשה.

3. סוגים שונים של ViewResolvers

Spring MVC היא מסגרת MVC Web לבניית יישומי אינטרנט. באופן כללי כל מסגרות ה-MVC מספקות דרך לעבוד עם תצוגות. Spring עושה זאת באמצעות ה-ViewResolvers, המאפשרים לך להציג דגמים בדפדפן מבלי לקשור את היישום לטכנולוגיית תצוגה ספציפית. עכשיו בואו נבין את ViewResolver עם פרויקט לדוגמה ב-STS.

המסגרת של Spring MVC מורכבת מהרכיבים הבאים:

Model: מודל יכול להיות אובייקט או אוסף של אובייקטים שמכיל בעצם את הנתונים של האפליקציה.

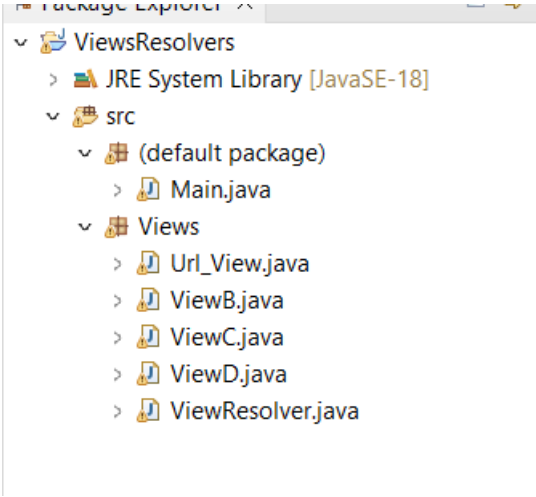
View: תצוגה משמשת להצגת המידע למשתמש בפורמט מסוים. האביב תומך בטכנולוגיות שונות כמו סמן חופשי, מהירות ו-thymeleaf.

Controller: הוא מכיל את החלק ההגיוני של האפליקציה. הערת @Controller משמשת לסימון מחלקה זו כבקר.

FrontController: הוא נשאר אחראי על ניהול הזרימה של יישום האינטרנט. DispatcherServlet פועל כבקר קדמי ב- Spring MVC.

ViewResolvers	הגדרה
InternalResourceViewResolver	ה-InternalResourceViewResolver משמש כדי לפתור את ה-URI שסופק ל-URI בפועל. הדוגמה הבאה מראה כיצד להשתמש ב-InternalResourceViewResolver באמצעות Spring Web MVC Framework. ה-InternalResourceViewResolver מאפשר מיפוי דפי אינטרנט עם בקשות.
UrlBasedViewResolver	UrlBasedViewResolver הוא היישום של ממשק ViewResolver. זה עובד על בסיס שם תצוגה סמלי. כאשר שיטה במחלקת בקר מפנה לדף, זה עוזר לגלות את נתיב הקובץ בפועל.

ResourceBundleViewResolver	ה-ResourceBundleViewResolver משמש כדי לפתור את שמות התצוגה באמצעות שעועית תצוגה המוגדרת בקובץ המאפיינים. הדוגמה הבאה מראה כיצד להשתמש ב-ResourceBundleViewResolver באמצעות Spring Web MVC Framework.
XmlViewResolver	ה-XmlViewResolver משמש כדי לפתור את שמות התצוגה באמצעות שעועית תצוגה המוגדרת בקובץ xml. הדוגמה הבאה מראה כיצד להשתמש ב-XmlViewResolver באמצעות Spring Web MVC framework.
TilesViewResolver	Spring מספקת תמיכה באינטגרציה עם מסגרת אריחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את הפריסה של אפליקציית Spring MVC בעזרת תמיכת אריחי קפיצים. היתרון של תמיכת אריחים ב- Spring MVC **שימוש חוזר:** אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. **שליטה מרכזית:** אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. **קל לשנות את הפריסה:** בעזרת דף תבנית בודד, נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery וכו'.
BeanNameViewResolver	BeanNameViewResolver ממפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי DispatcherServlet כדי לעבד את התצוגה.




```java
import java.util.Arrays;

public class Main {

    public static void main(String[] args) {

        Url_View url = new Url_View();

    }

}

```

```java
package Views;

import java.awt.Color;

public class Url_View extends JFrame{

    private JPanel panel;
    private JLabel label;
    private JTextField jtf;

    public Url_View() {
        this.setTitle("ViewA");
        this.setSize(500, 500);
        this.setLocationRelativeTo(null);
        label = new JLabel();
        Font font = new Font("Arial",Font.BOLD,30);
        label.setText("Enter your url please: ");
        label.setFont(font);
        label.setBounds(100,100,400,50);
        jtf = new JTextField();
        jtf.setSize(300, 50);
        jtf.setLocation(100, 155);
        jtf.setFont(font);
        jtf.addKeyListener(new ButtonListener());

        panel = new JPanel();
        panel.setLayout(null);
        panel.setBackground(Color.GREEN);
        panel.add(label);
        panel.add(jtf);
        this.setContentPane(panel);
```

```java
        this.setVisible(true);
    }

    class ButtonListener implements KeyListener{


        @Override
        public void keyTyped(KeyEvent e) {


        }

        @Override
        public void keyPressed(KeyEvent e) {
            if(e.getKeyCode() == KeyEvent.VK_ENTER) {
                ViewResolver vr= new ViewResolver(jtf.getText().toString());
                String cypher = vr.cryptoDatas();
                vr.designView(cypher);
            }
        }

        @Override
        public void keyReleased(KeyEvent e) {


        }

    }

}
```

```java
package Views;

import java.awt.Color;

public class ViewB extends JFrame{

    private JPanel panel;
    private JLabel label;

    public ViewB() {
        this.setTitle("ViewB");
        this.setSize(500, 500);
        this.setLocationRelativeTo(null);
        label = new JLabel();
        label.setText("ViewB");
        label.setBounds(100,100,100,100);

        panel = new JPanel();
        panel.setLayout(null);
        panel.setBackground(Color.YELLOW);
        panel.add(label);
        this.setContentPane(panel);
        this.setVisible(true);
    }

}
```

```java
package Views;

import java.awt.Color;

public class ViewC extends JFrame{

    private JPanel panel;
    private JLabel label;

    public ViewC() {
        this.setTitle("ViewC");
        this.setSize(500, 500);
        this.setLocationRelativeTo(null);
        label = new JLabel();
        label.setText("ViewC");
        label.setBounds(100,100,100,100);

        panel = new JPanel();
        panel.setLayout(null);
        panel.setBackground(Color.RED);
        panel.add(label);
        this.setContentPane(panel);
        this.setVisible(true);
    }

}
```

```java
package Views;

import java.awt.Color;

public class ViewD extends JFrame{

    private JPanel panel;
    private JLabel label;

    public ViewD() {
        this.setTitle("ViewD");
        this.setSize(500, 500);
        this.setLocationRelativeTo(null);
        label = new JLabel();
        label.setText("ViewD");
        label.setBounds(100,100,100,100);

        panel = new JPanel();
        panel.setLayout(null);
        panel.setBackground(Color.BLUE);
        panel.add(label);
        this.setContentPane(panel);
        this.setVisible(true);
    }

}
```

```java
package Views;

import java.util.regex.Matcher;

public class ViewResolver {

    private static final String REGEX1 = "[*$]";
    private static final String REGEX2 = "[10]";
    private String url;



    public ViewResolver(String url) {
        this.url=url;
    }

```

```java
public String cryptoDatas() {

    String result="";
    char cyptherLetter;
    for(int i=0;i<this.url.length();i++) {
            if(this.url.charAt(i) >=65 && this.url.charAt(i) <=72 ) {
                cyptherLetter = '*';
                result+=cyptherLetter;
            }
            else if(this.url.charAt(i) >=73 && this.url.charAt(i) <=80 ) {
                cyptherLetter = '$';
                result+=cyptherLetter;
            }
            else if(this.url.charAt(i) >=81 && this.url.charAt(i) <=90 ) {
                cyptherLetter = '1';
                result+=cyptherLetter;

            }else {
                cyptherLetter = '0';
                result+=cyptherLetter;
            }
        }


        return result;

}
```

```java
    public void designView(String cyphertext) {
        Pattern p1 = Pattern.compile(REGEX1);
        Pattern p2 = Pattern.compile(REGEX2);

        Matcher m1 = p1.matcher(cyphertext);
        Matcher m2 = p2.matcher(cyphertext);

        if(m1.find()) {
            ViewB v = new ViewB();

        }
        else if(m2.find()) {
            ViewC v = new ViewC();

        }
        else {
            ViewD v = new ViewD();

        }




    }
```