JAVA + SPRING MVC

ViewResolvers :04 שיעור

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 אשר חייב להכיל את המטפל שתואם את הבקשה, 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

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

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.

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

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

ResourceBundleViewResolver - לפתור את שמות התצוגה באמצעות שעועית תצוגה המאצעות שעועית תצוגה המאצעות שעועית תצוגה המאצעות שעועית תצוגה באמצעות שעועית תצוגה באמצעות שעועית הדא מראה מראה מראה ביצד להשתמש ב- ביצד להשתמש ב- ב- Spring Web MVC Framework הרצוא באמצעות שעועית תצוגה המאבדת הצא שמות התצוגה המאצעות שעועית תצוגה המוגדרת בקובץ Image של TilesViewResolver בקובץ Image מספקת תתיכה באה מראה כיצד להשתמש ב- MVC framework ב- MVC framework בארצעות ששועית מטגרת אריחי אפאציי. אז אנחנו כילוים פשוט לנהל את Spring MVC בעזרת אריחי אפאצי. אז אנחנו כילוים פשוט לנהל את Spring MVC בעזרת הפריסה של אפליקציית Spring MVC בעזרת שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בלבד. על ידי עמוד תבנית בודד בלבד. שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר קל לשנות את פריסת העמוד בכל עת. כך שהאתר קל לשנות את פריסת העמוד בכל עת. כך שהאתר קל לשנות את פריסת העמוד בכל עת. כך שהאתר קל לשנות את פריסת העמוד בכל עת. כך שהאתר הבשום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית בולן יהים את ממשק ה-שוט Cipy לייבי אורעים על עיביע ועליים את ממשק ה-שועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית ברקול ועל-ידי BeanNameViewResolver שניתן יהיה להפעילו על-ידי DispatcherServlet כדי לעבד את התצוגה.		
המוגדרת בקובץ המאפיינים. הדוגמה הבאה מראה בצור מוגד או רייבו להשתמש ב- ResourceBundleViewResolver Spring Web MVC Framework בא XmlViewResolver או	ה-ResourceBundleViewResolver משמש כדי	ResourceBundleViewResolver
ריצד להשתמש ב- Spring Web MVC Framework	לפתור את שמות התצוגה באמצעות שעועית תצוגה	
ResourceBundleViewResolver .Spring Web MVC Framework XmiViewResolver -Spring Web MVC Framework XmiViewResolver Audit Analysis and architecture Spring Web with a feature and and architecture Equipped More and a feature Spring Web בקובץ IViewResolver בקובץ MVC framework TilesViewResolver ארוחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את ארוחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את Aprin אפרוחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את Spring MVC בעזרת היתרון של תמיכת אריחים ב- Spring MVC שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה שימוש חוזר ברכיב ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בלד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד לבלד. שליטה את פריסת העמוד בכל עת. כך שהאתר קל לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו (כל לשנות את פריסת העמוד בכל עת. כך שהאתר שם bootstrap, jQuery BeanNameViewResolver התואמת חייבת ליישם את ממשק ה-waiv Cpipped (א-ידי ויהיה להפעילו על-ידי שליתו את הפעילו על-ידי שנועועית בהקשר היישום הנוכחי. השעועית DispatcherServlet שנותן יהיה להפעילו על-ידי denury בדי בל מתוד לחיבות שליש את ממשק ה-waiv Cpipped (א-ידי שויים) אור בעורית בהקשר היישום הנוכחי. השעועית DispatcherServlet שניתן יהיה להפעילו על-ידי weiver and the manufacture and the	המוגדרת בקובץ המאפיינים. הדוגמה הבאה מראה	
Spring Web MVC Framework ה-Charlem Country (August of Spring Web Wilson) און אינו אינו אינו אינו אינו אינו אינו אינו	-כיצד להשתמש ב	
שמות התצוגה באמצעות שעועית תצוגה המוגדרת שמות התצוגה המוגדרת מצוגה המוגדרת בקבץ וואד. הדוגמה הבאה מראה כיצד להשתמש Spring Web באמצעות שעועית תצוגה המוגדרת באמצעות שעועית תצוגה המוגדרת באמצעות שעועית תצוגה המוגדרת MVC framework באמצעות שמסגרת MVC framework מספקת תמיכה באינטגרציה עם מסגרת אריחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את Spring MVC בעזרת המיכת אריחי קפיצים. Spring MVC 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. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בל עת. כך שהאתר של לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלים bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום את ממשק ה-webl כך לשם עועית בהקשר היישום את ממשק ה-webl כך לשנית יהה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי Dispatcher Servlet שניתן יהיה להפעילו על-ידי שום Constrap ועדר בתוצוגה Dispatcher Servlet שניתן יהיה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי שניתן יהיה להפעילו על-ידי של Dispatcher Servlet שניתן יהיה להפעילו על-ידי שניתן יהיה להפעילו על-ידי שניתן יהיה להפעילו על-ידי של Dispatcher Servlet בעדר באום בדים להשתמש של ביד לשלום בערים בעובד לבת ביד להשתמים ביד להשתמש של ביד לביד להשתמש של ביד להשתמש של ביד להשתמה של ביד להשתמש של ביד לביד להשתמש של ביד להשתמה של ביד להשתמש של ביד להשתמה של ביד להשתמה של ביד להשתמה של ביד להשתמש של ביד להשתמה של ביד להשתמש של ביד להשתמה של ביד להשתמה של ביד להשתמש של ביד להשתמה של ביד להשתמש של ביד להשתמה של ביד להשתמש של ביד להשתמה	באמצעות ResourceBundleViewResolver	
שמות התצוגה באמצעות שעועית תצוגה המוגדרת בקובץ אוצר. הדוגמה הבאה מראה כיצד להשתמש בקובץ xmlViewResolver. בקובץ xmlViewResolver. בקובץ xmlViewResolver באמצעות שמט אוצר. אוצריביה עם מסגרת MVC framework מספקת תמיכה באינטגרציה עם מסגרת אריחי אפאצ'י. אז אנחנו יכולים פשוט לנהל את הפריסה של אפליקציית Spring MVC בעזרת המיכת אריחים ב- Spring MVC בעזרת היתרון של תמיכת אריחים ב- Spring MVC שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בלבד. שלך יכול לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית DispatcherServlet על שניתן יהיה להפעילו על-ידי dong the form of the the for	.Spring Web MVC Framework	
בקובץ אווא. הדוגמה הבאה מראה כיצד להשתמש ב-Spring Web באמצעות XmlViewResolver ב-mWC framework Spring Nocard Spring MVC הפריסה של אפליקציית Spring MVC בעזרת היתרון של תמיכת אריחי קפיצים. Spring MVC היתרון של תמיכת אריחים ב- Spring MVC שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה בודד במספר דפים כמו רכיבי כותרת עליונה שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד נוכל לשנות את פריסה: בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו וסוס bootstrap, jQuery BeanNameViewResolver שניתן יהיה להפעילו על-ידי BorankameViewResolver שניתן יהיה להפעילו על-ידי DispatcherServlet	ה-XmlViewResolver משמש כדי לפתור את	XmlViewResolver
ארויי אפאציות מיכה באינטגרציה עם מסגרת Spring מספקת תמיכה באינטגרציה עם מסגרת אריחי אפאציי. אז אנחנו כולים פשוט לנהל את אריחי אפאציי. אז אנחנו כולים פשוט לנהל את Spring MVC בעזרת תמיכת אריחי קפיצים. Spring MVC - תמיכת אריחים ב- Spring MVC בודד במספר דפים כמו רכיבי כותרת עליונה שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד ברכיב על ידי עמוד תבנית בודד בלבד. של לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמן אקל בעלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמן בקלות טכנולוגיות חדשות כמו וכוס bootstrap, jQuery BeanNameViewResolver ממפה שם תצוגה התואמת חייבת ליישם את ממשק ה-Will DispatcherServlet שניתן יהיה להפעילו על-ידי View-DispatcherServlet	שמות התצוגה באמצעות שעועית תצוגה המוגדרת	
ארויי אפאציות מיכה באינטגרציה עם מסגרת Spring מספקת תמיכה באינטגרציה עם מסגרת אריחי אפאציי. אז אנחנו כולים פשוט לנהל את אריחי אפאציי. אז אנחנו כולים פשוט לנהל את Spring MVC בעזרת תמיכת אריחי קפיצים. Spring MVC - תמיכת אריחים ב- Spring MVC בודד במספר דפים כמו רכיבי כותרת עליונה שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד ברכיב על ידי עמוד תבנית בודד בלבד. של לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמן אקל בעלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמן בקלות טכנולוגיות חדשות כמו וכוס bootstrap, jQuery BeanNameViewResolver ממפה שם תצוגה התואמת חייבת ליישם את ממשק ה-Will DispatcherServlet שניתן יהיה להפעילו על-ידי View-DispatcherServlet	בקובץ xml. הדוגמה הבאה מראה כיצד להשתמש	
אריחי אפאצ"י. אז אנחנו יכולים פשוט לנהל את אריחי אפאצ"י. אז אנחנו יכולים פשוט לנהל את אריחי אפאצ"י. אז אנחנו יכולים פשוט לנהל את הפריסה של אפליקציית Spring MVC בעזרת תמיכת אריחי קפיצים. Spring MVC – שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד בלבד. על ידי עמוד תבנית בודד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו וbootstrap, jQuery BeanNameViewResolver שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום את ממשק ה-Niep על יהיה להפעילו על-ידי View-		
אריחי אפאצי ¹ . אז אנחנו יכולים פשוט לנהל את הפריסה של אפליקציית Spring MVC עזרת הפריסה של אפליקציית קיצים. מימיכת אריחי קפיצים. מימיכת אריחים ב- Spring MVC שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לשנות את פריסה מנולוגיות חדשות כמו נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery שלי יכול לשנות בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית בהקשר היישם את ממשק ה-DispatcherServlet שניתן יהיה להפעילו על-ידי doublesservlet שניתן יהיה להפעילו על-ידי Misamu View-Resolver		
אריחי אפאצי ¹ . אז אנחנו יכולים פשוט לנהל את הפריסה של אפליקציית Spring MVC עזרת הפריסה של אפליקציית Spring MVC תמיכת אריחי קפיצים. שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, פול לשנות את פריסה בעזרת דף תבנית בודד, שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו נוכל לשנות את פריסה ממפה שם תצוגה bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית DispatcherServlet שניתן יהיה להפעילו על-ידי doublespatcherServlet שניתן יהיה להפעילו על-ידי שורחים.	מספקת תמיכה באינטגרציה עם מסגרת Spring	TilesViewResolver
תמיכת אריחי קפיצים. Spring MVC - היתרון של תמיכת אריחים ב- שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, וכל לשנות את הפריסה העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית DispatcherServlet על-ידי View-		
תמיכת אריחי קפיצים. Spring MVC - היתרון של תמיכת אריחים ב- שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, וכל לשנות את הפריסה העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית DispatcherServlet על-ידי View-	בעזרת Spring MVC בעזרת	
אימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב בודד במספר דפים כמו רכיבי כותרת עליונה ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד על ידי עמוד תבנית בודד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, שלך יכול לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver ממפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית DispatcherServlet על-ידי View- שניתן יהיה להפעילו על-ידי	, ,	
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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View את ממשק ה-View שניתן יהיה להפעילו על-ידי View שורים והיה להפעילו על-ידי	'	
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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View את ממשק ה-View שניתן יהיה להפעילו על-ידי View שורים והיה להפעילו על-ידי	Spring MVC -היתרון של תמיכת אריחים ב	
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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver ממפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View את ממשק ה-View שניתן יהיה להפעילו על-ידי DispatcherServlet		
ותחתית. שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד 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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View- סניתן יהיה להפעילו על-ידי DispatcherServlet	שימוש חוזר: אנו יכולים לעשות שימוש חוזר ברכיב	
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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View- DispatcherServlet שניתן יהיה להפעילו על-ידי	בודד במספר דפים כמו רכיבי כותרת עליונה	
על ידי עמוד תבנית בודד בלבד. קל לשנות את הפריסה: בעזרת דף תבנית בודד, נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו טנולוגיות חדשות כמו וכו'. bootstrap, jQuery BeanNameViewResolver מפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View לו על-ידי DispatcherServlet	ותחתית.	
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 שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery וכו'. bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View את ממשק ה-View שניתן יהיה להפעילו על-ידי DispatcherServlet		
קל לשנות את הפריסה: בעזרת דף תבנית בודד, נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו bootstrap, jQuery .'io bootstrap, jQuery BeanNameViewResolver מפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי DispatcherServlet	שליטה מרכזית: אנו יכולים לשלוט בפריסת העמוד	
נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו טכול לאמץ בקלות טכנולוגיות חדשות כמו וכו'. bootstrap, jQuery הפה שם תצוגה BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי	על ידי עמוד תבנית בודד בלבד.	
נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו טכול לאמץ בקלות טכנולוגיות חדשות כמו וכו'. bootstrap, jQuery הפה שם תצוגה BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי		
נוכל לשנות את פריסת העמוד בכל עת. כך שהאתר שלך יכול לאמץ בקלות טכנולוגיות חדשות כמו טכול לאמץ בקלות טכנולוגיות חדשות כמו וכו'. bootstrap, jQuery הפה שם תצוגה BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי	קל לשנות את הפריסה: בעזרת דף תבנית בודד,	
שלך יכול לאמץ בקלות טכנולוגיות חדשׂות כמו bootstrap, jQuery בוי bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית העועית בהקשר היישום הנוכחי. השעועית עופאר היישום הנוכחי. השעועית בהקשר היישום הנוכחי. השעועית טופאר היישום את ממשק ה-View שניתן יהיה להפעילו על-ידי DispatcherServlet	-	
וכו'. bootstrap, jQuery BeanNameViewResolver לשם שעועית בהקשר היישום הנוכחי. השעועית לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View שניתן יהיה להפעילו על-ידי DispatcherServlet	•	
ממפה שם תצוגה BeanNameViewResolver ממפה שם תצוגה לשם שעועית בהקשר היישום הנוכחי. השעועית התואמת חייבת ליישם את ממשק ה-View כך שניתן יהיה להפעילו על-ידי DispatcherServlet		
כך View-התואמת חייבת ליישם את ממשק DispatcherServlet שניתן יהיה להפעילו על-ידי		BeanNameViewResolver
DispatcherServlet שניתן יהיה להפעילו על-ידי	לשם שעועית בהקשר היישום הנוכחי. השעועית	
·	התואמת חייבת ליישם את ממשק ה-View כך	
כדי לעבד את התצוגה.		
	כדי לעבד את התצוגה.	

```
ViewsResolvers

> ■ JRE System Library [JavaSE-18]

> □ src

- □ (default package)

- □ Main.java

- □ Views

- □ Url_View.java

- □ ViewB.java

- □ ViewD.java

- □ ViewD.java

- □ ViewD.java
```

```
1* import java.util.Arrays;
5
6
public class Main {
7
8** public static void main(String[] args) {
9
Url_View url = new Url_View();
11
12  }
13
14 }
15
```

```
1 package Views;
% 3*import java.awt.Color; ...
14
№15 public class Url View extends JFrame{
       private JPanel panel;
private JLabel label;
17
 18
 19
        private JTextField jtf;
 20
 21⊖
        public Url_View() {
 22
            this.setTitle("ViewA");
 23
            this.setSize(500, 500);
 24
            this.setLocationRelativeTo(null);
 25
            label = new JLabel();
            Font font = new Font("Arial", Font.BOLD, 30);
 26
 27
            label.setText("Enter your url please: ");
 28
            label.setFont(font);
 29
            label.setBounds(100,100,400,50);
 30
            jtf = new JTextField();
 31
            jtf.setSize(300, 50);
 32
            jtf.setLocation(100, 155);
 33
            jtf.setFont(font);
 34
            jtf.addKeyListener(new ButtonListener());
 3.5
 36
            panel = new JPanel();
 37
            panel.setLayout(null);
 38
            panel.setBackground(Color. GREEN);
 39
            panel.add(label);
 40
            panel.add(jtf);
41
            this.setContentPane(panel);
```

```
42
            this.setVisible(true);
43
        }
44
45⊖
        class ButtonListener implements KeyListener{
46
47
48⊖
            @Override
△49
            public void keyTyped(KeyEvent e) {
50
51
52
            }
53
54⊜
            @Override
△55
            public void keyPressed(KeyEvent e) {
56
                if(e.getKeyCode() == KeyEvent.VK ENTER) {
57
                    ViewResolver vr= new ViewResolver(jtf.getText().toString());
                    String cypher = vr.cryptoDatas();
58
59
                    vr.designView(cypher);
60
                }
61
            }
62
63⊜
            @Override
△64
            public void keyReleased(KeyEvent e) {
65
66
67
            }
68
69
        }
70
71 }
```

```
1 package Views;
 2
 3⊕ import java.awt.Color; ...
9 public class ViewB extends JFrame {
10
11
        private JPanel panel;
12
        private JLabel label;
13
14⊖
        public ViewB() {
15
            this.setTitle("ViewB");
16
            this.setSize(500, 500);
17
            this.setLocationRelativeTo(null);
18
            label = new JLabel();
19
            label.setText("ViewB");
20
            label.setBounds(100,100,100,100);
21
22
            panel = new JPanel();
23
            panel.setLayout(null);
24
            panel.setBackground(Color.YELLOW);
25
            panel.add(label);
26
            this.setContentPane(panel);
27
            this.setVisible(true);
28
        }
29
30 }
31
32
```

```
1 package Views;
 2
 3⊕import java.awt.Color;
9 public class ViewC extends JFrame {
10
11
        private JPanel panel;
12
        private JLabel label;
13
14⊖
        public ViewC() {
15
            this.setTitle("ViewC");
16
            this.setSize(500, 500);
17
            this.setLocationRelativeTo(null);
18
            label = new JLabel();
19
            label.setText("ViewC");
20
            label.setBounds(100,100,100,100);
21
22
            panel = new JPanel();
23
            panel.setLayout(null);
24
            panel.setBackground(Color.RED);
25
            panel.add(label);
26
            this.setContentPane(panel);
27
            this.setVisible(true);
28
        }
29
30 }
21
```

```
1 package Views;
 3⊕ import java.awt.Color; ...
9 public class ViewD extends JFrame{
10
11
        private JPanel panel;
12
        private JLabel label;
13
14⊖
        public ViewD() {
15
            this.setTitle("ViewD");
16
            this.setSize(500, 500);
17
            this.setLocationRelativeTo(null);
18
            label = new JLabel();
19
            label.setText("ViewD");
20
            label.setBounds(100,100,100,100);
21
22
            panel = new JPanel();
23
            panel.setLayout(null);
24
            panel.setBackground(Color.BLUE);
25
            panel.add(label);
26
            this.setContentPane(panel);
27
            this.setVisible(true);
28
        }
29
30 }
 31
32
33
```

```
1 package Views;
3⊕import java.util.regex.Matcher;...
 8 public class ViewResolver {
 9
10
       private static final String REGEX1 = "[*$]";
11
       private static final String REGEX2 = "[10]";
12
       private String url;
13
14
15
16⊜
       public ViewResolver(String url) {
17
           this.url=url;
18
19
```

```
public String cryptoDatas() {
        String result="";
        char cyptherLetter;
        for(int i=0;i<this.url.length();i++) {</pre>
               if(this.url.charAt(i) >=65 && this.url.charAt(i) <=72 ) {</pre>
                   cyptherLetter = '*';
                   result+=cyptherLetter;
               else if(this.url.charAt(i) >=73 && this.url.charAt(i) <=80 ) {</pre>
                   cyptherLetter = '$';
                   result+=cyptherLetter;
               else if(this.url.charAt(i) >=81 && this.url.charAt(i) <=90 ) {</pre>
                   cyptherLetter = '1';
                   result+=cyptherLetter;
               }else {
                   cyptherLetter = '0';
                   result+=cyptherLetter;
               }
           }
           return result;
    }
10
50⊝
        public void designView(String cyphertext) {
51
             Pattern p1 = Pattern.compile(REGEX1);
52
             Pattern p2 = Pattern.compile(REGEX2);
53
54
             Matcher m1 = p1.matcher(cyphertext);
55
             Matcher m2 = p2.matcher(cyphertext);
56
57
             if(m1.find()) {
58
                  ViewB v = new ViewB();
59
60
61
             else if(m2.find()) {
62
                  ViewC v = new ViewC();
63
64
65
             else {
66
                  ViewD v = new ViewD();
67
68
             }
69
70
71
72
73
        }
74
75
```