



J2EE 6 vs Spring / Back to Even?

Joel Tosi

Senior Solutions Architect
jtosi@redhat.com

Who / Why

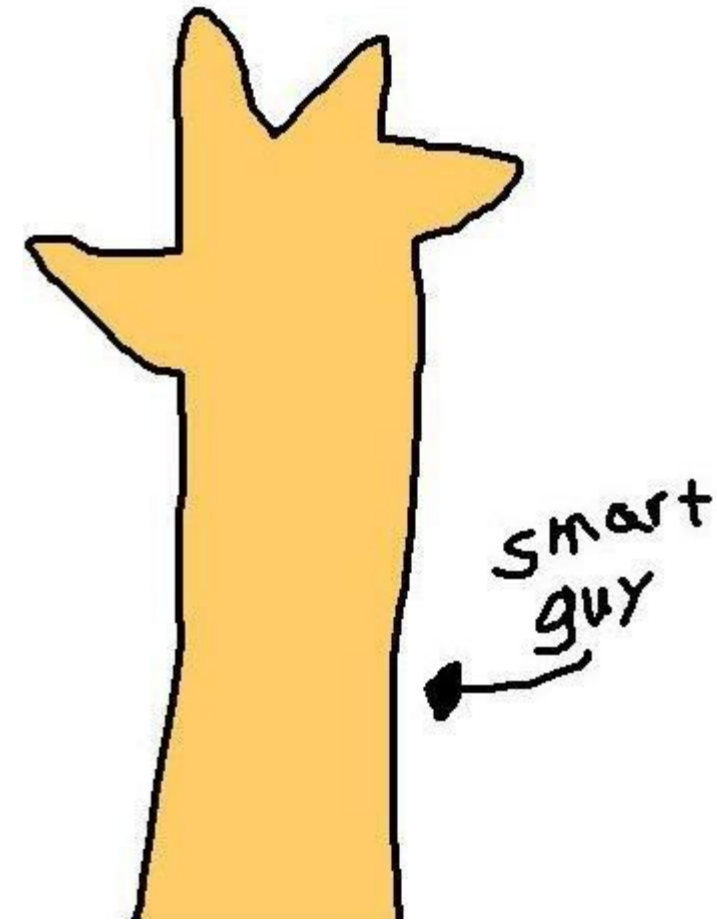
Who uses Spring?

Why do you use Spring?

Who uses / has used J2EEN?

Why did you use J2EEN?

Which is more innovative?



Myths



Topics

Development

Testing

Deployments

Comparatively Speaking

Feature	Spring	J2EE6
Web Framework	Spring MVC	JSF / EJB (CDI) / JPA
Data Access	JPA, Template	JPA
Dependency Injection	DI	CDI
Restful Services	Spring MVC	JAX-RS
Integration Testing	Spring Test	Arquillian

Say I want to build a web app...

```
@Entity
```

```
public class Recipe
```

```
{
```

```
    @Id
```

```
    Long recipeld;
```

```
    String name;
```

```
    List <Ingredient> ingredients;
```

Business Logic

@Service

@Transactional

```
public class RecipeFacadeBean implements RecipeFacade
```

```
{
```

vs

@Stateless

```
public class RecipeFacadeBean implements RecipeFacade
```

```
{
```

View / Controller - Spring

```
<form method="post" action="/recipe/new">
```

```
...
```

```
    <input type="submit" value="Create">
```

```
</form>
```

```
@RequestMapping (value="/recipe/new", method=RequestMethod.POST)
```

```
public class Recipe Controller {
```

```
    public String createNewRecipePost(...
```

View Controller - J2EE

```
<h:form>
```

```
...
```

```
    <h:commandButton value="Create"  
    action="#{recipeController.createNewRecipePost}" />
```

```
</h:form>
```

```
@Named
```

```
public class RecipeController
```

```
{
```

```
    public String createNewRecipePost(...
```

View Rendering - Spring

```
@RequestMapping (value="/recipe/new", method=RequestMethod.POST)
```

```
public class Recipe Controller {
```

```
    public String createNewRecipePost(...
```

```
{
```

```
    model.addAttribute("recipe", recipe);
```

```
    return "/user/confirmation";
```

/user/confirmation.jsp

Recipe Name: #{recipe.name}

Ingredients: #{recipe.ingredients}

View Rendering - J2EE

@Named

```
public class RecipeController
{
    Recipe recipe = new Recipe();
    public String createNewRecipePost(...)
    {
        return "recipe_confirm";
    }
}
```

/user/confirmation.jsp

Recipe Name: #{recipeController.recipe.name}

Ingredients: #{recipeController.recipe.ingredients}

Passing State – Spring

```
@RequestMapping (value="/recipe/new", method=RequestMethod.POST)
```

```
public class Recipe Controller {
```

```
    public String createNewRecipePost(...
```

```
{
```

```
    model.addAttribute("recipe", recipe);
```

```
    return "/user/confirmation";
```

```
}
```

```
@RequestMapping (value="/recipe/confirm",  
method=RequestMethod.POST)
```

```
public String confirmNewRecipe(Recipe....., Model.....)
```

```
{
```

```
    rf.persist(recipe);
```

```
}
```

Passing State – Spring

/user/confirmation.jsp

```
<form:form modelAttribute="recipe" method="post"  
action="recipe_confirm">
```

```
    Recipe Name: #{recipe.name}
```

```
    Ingredients: #{recipe.ingredients}
```

```
    <input type="submit" value="For sure" />
```

```
    <form:hidden path="name"/>
```

```
    <form:hidden path="ingredients"/>
```

```
</form:form>
```

Passing State - J2EE

@Named

```
public class RecipeController
{
    Recipe recipe = new Recipe();

    public String createNewRecipePost()
    {
        conversation.begin();

        return "recipe_confirm";
    }

    public String confirmNewRecipe()
    {
        rf.persist(recipe);

        conversation.end();
    }
}
```

Passing State - J2EE

/user/confirmation.jsp

```
<h:form>
```

```
    Recipe Name: #{recipeController.recipe.name}
```

```
    Ingredients: #{recipeController.recipe.ingredients}
```

```
    <h:commandButton action="#" #{recipeController.confirmNewRecipe}"  
    value = "For sure"/>
```

```
</h:form>
```

Validation - Spring

```
public String confirmNewRecipe(@Valid Recipe recipe, BindingResult
results, Model model)
{
    if(results.hasErrors())
    {
        doSomething();
    }

    rf.persist(recipe);
}
```

Validation - J2EE

Testing

DI vs CDI

Integration Testing

DI vs CDI

CDI is DI ++

```
public class PriceMessenger {  
  
    @Inject Event<PriceEvent> priceEvents;  
  
    public void hello(){  
  
        priceEvents.fire(new PriceEvent(...));  
  
    }  
  
}  
  
public class PriceListener {  
  
    public void listenToPrices(@Observes PriceEvent priceEvent){  
  
    ...  
  
}
```

Integration Testing

Testing your app live without Mocks

Deployments

With Spring, developers will own and manage their dependencies

In EE6, middleware services are deferred to the EE provider

What is better?

Conclusion



Questions

NEVER HAVE I FELT SO
CLOSE TO ANOTHER SOUL
AND YET SO HELPLESSLY ALONE
AS WHEN I GOOGLE AN ERROR
AND THERE'S ONE RESULT
A THREAD BY SOMEONE
WITH THE SAME PROBLEM
AND NO ANSWER
LAST POSTED TO IN 2003



jtosi@redhat.com
@joeltosi



Questions?