

13 JSP Standard Tag Library

- 13.1 Bibliotek med nya JSP-kommandon
- 13.2 JSP Standard Tag Library (JSTL)
- 13.3 Filstruktur för webbapplikationer med JSTL
- 13.4 Deklaration av JSP-kommandon
- 13.5 Lägga till biblioteksfiler för JSTL
- 13.6 Kommandobibliotek i JSTL 1.2

PROV

PROV

Bibliotek med nya JSP-kommandon

JSP är inte begränsat till den uppsättning JSP-kommandon som ingår initialt.

Det är möjligt att utveckla *kommandobibliotek* med nya JSP-kommandon.

(Engelska termen är *custom tag libraries*.)

Utveckling sker genom deklARATIONER i XML och programmering i Java.

Exempel på tänkbara kommandobibliotek:

- Utbyggnad med nya språkelement
- Förenklad åtkomst till JDBC och JNDI
- Inkapsling av åtkomst till affärssystem
- Integration med modellklasser i Swing
- Diagramgrafik

Vi har redan visat hur JSP-kommandon gör det lättare att integrera JSP med JavaBeans-komponenter. JSP är emellertid inte begränsat till den uppsättning av JSP-kommandon som ingår initialt, utan det är möjligt att utveckla *kommandobibliotek* (den engelska termen är *custom tag libraries*) bestående av nya JSP-kommandon med gemensamt prefix för biblioteket. Sådana JSP-kommandon beskrivs deklarativt i en XML-beskrivning, och kan stödjas av exekverbar kod utvecklad i Java. Utveckling av sådana kommandobibliotek ligger emellertid helt utanför ramen för denna kurs.

Det finns många tänkbara tillämpningar av kommandobibliotek. Några exempel är:

- utbyggnad med nya språkelement för att reducera kodvolymen, t ex iteration över datastrukturer
- förenklad åtkomst till ofta använda delar av standardbiblioteket, som exempelvis **JDBC** och **JNDI**
- inkapsling av åtkomst till affärssystem som inte JSP-utvecklare bör ha obegränsad tillgång till
- integration av JSP med de mer avancerade modellklasserna i Javas användargränssnittsarkitektur **Swing**, t ex `ListModel` och `TableModel`
- stöd för att skapa diagramgrafik

JSP Standard Tag Library (JSTL)

Fr o m JSP 1.2 finns ett standardbibliotek, **JSP Standard Tag Library (JSTL)**:

- Innehåller JSP-kommandon för styrstrukturer, nyttofunktioner m m
- JSP-direktivet `taglib` binder ett lokalt prefix till varje bibliotek
- Varje bibliotek har ett standardprefix; kan ändras för att undvika kollisioner
- Attributvärden kan innehålla JSP-uttryck

```
<!-- Dice JSP - Lennart Månsson - 2007-10-14 -->
<%@ page info="Dice Throw JSP - Version 1.1" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

...
    int nDice = 0;
    if (count != null && count.length() > 0)
        nDice = Integer.parseInt(count);
%>
<HR><%= player %>, dina slag:
<c:forEach var="i" begin="1" end="<%= nDice %>">
    &nbsp;&nbsp;&nbsp;<%= 1 + (int)(6 * Math.random()) %>
</c:forEach>
</BODY>
</HTML>
```

Att förstärka JSP med bibliotek av nya JSP-kommandon är en så central idé att det fr o m JSP 1.2 finns ett standardbibliotek av JSP-kommandon, **JSP Standard Tag Library (JSTL)**. Syftet med JSTL är att bygga ut JSP inom ett antal centrala områden, för att på det sättet bredda den gemensamma bas som alla JSP-utvecklare har att utgå ifrån.

Bilden visar en variant av det tidigare exemplet med en JSP-sida för tärningsslag där Java-koden för en `for`-slinga som genererar upprepade tärningsslag har ersatts av JSP-kommandot `c:forEach` i JSTL. Det webbinnehåll som innesluts av märkorden `<c:forEach>` resp `</c:forEach>` kommer att genereras upprepade gånger. Antalet gånger styrs av kommandots attribut, där attributen `begin` och `end` anger start- resp slutvärde för en variabel som räknas upp med ett för varje nytt varv i slingan. Variabeln är åtkomlig i JSP under det variabelnamn som anges av attributet `var`:

```
<c:forEach var="i" begin="1" end="<%= nDice %>">
    &nbsp;&nbsp;&nbsp;<%= 1 + (int)(6 * Math.random()) %>
</c:forEach>
```

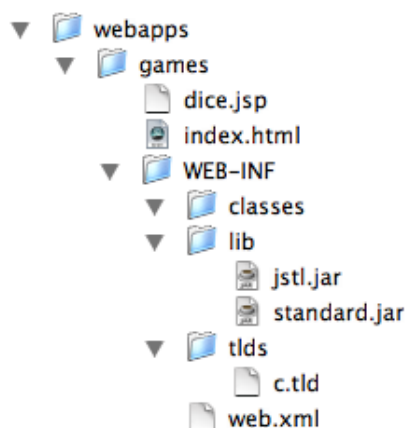
Notera att attributvärden precis som i tidigare fall kan vara dynamiska och anges av JSP-uttryck.

När ett kommandobibliotek ska användas – även de som ingår i JSTL – måste det först identifieras med hjälp av JSP-direktivet `<%@ taglib ... %>`, tex:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

där attributet `uri` anger en adress som unikt identifierar biblioteket, medan attributet `prefix` visar vilket lokalt prefix som ska användas för bibliotekets samtliga JSP-kommandon i den aktuella JSP-filen.

Filstruktur för webbapplikationer med JSTL

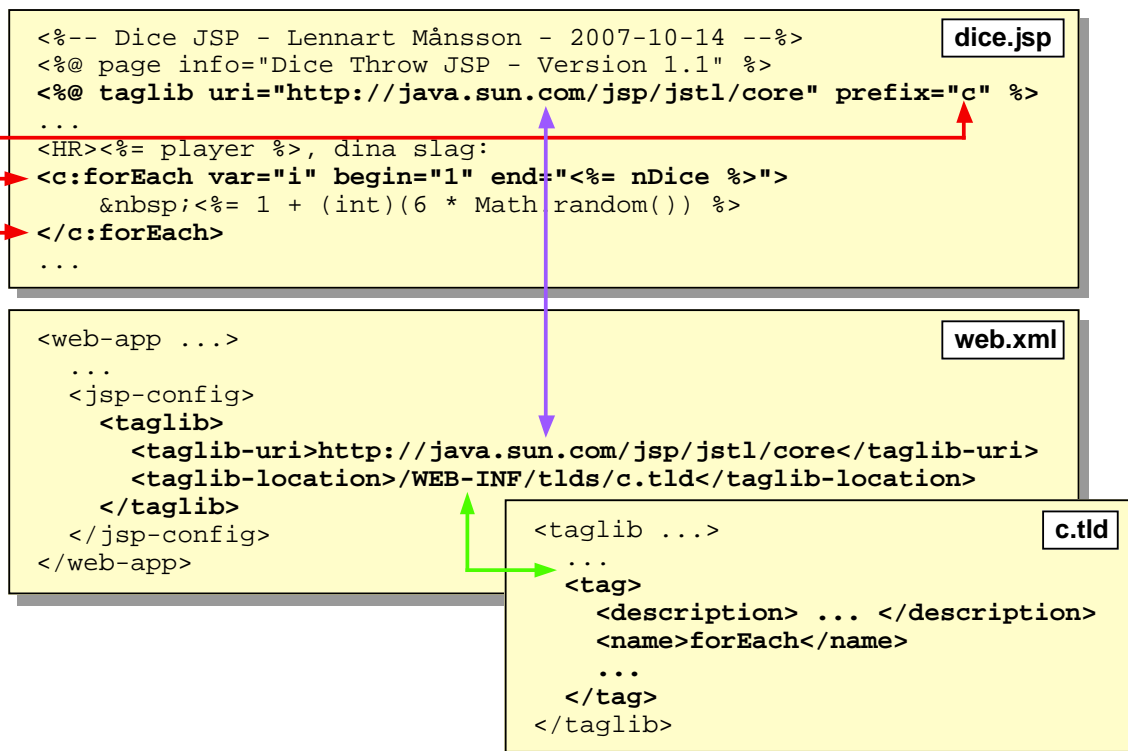


- JSTL består av två biblioteksfiler, `jstl.jar` och `standard.jar`
- JAR-filer för bibliotek placeras i underkatalogen `lib` till katalogen `WEB-INF`
- JSP-kommandon deklarereras i TLD-filer (*tag library descriptors*)
- TLD-filer placeras någonstans under katalogen `WEB-INF`
- TLD-filerna pekas ut av installationsbeskrivningen `web.xml`

En webbapplikation som utnyttjar JSP med JSTL följer samma övergripande filstruktur som vanliga webbapplikationer som utnyttjar enbart HTML och/eller JSP:

- Varje webbapplikation ges en egen katalog under den gemensamma rotkatalogen, med samma namn som webbapplikationen ges i URL:er. I bildens exempel har webbapplikationen namnet `games`.
- Filer med statiskt webbinnehåll som ska servas ut direkt till klienter utan ytterligare bearbetning, t ex HTML-filer, rena textfiler och bildfiler, placeras på den fysiska plats i filstrukturen som motsvarar den URL de adresseras med. Filen `index.html` placeras alltså direkt i webbapplikationens rotkatalog. Även JSP-filer som t ex `dice.jsp` placeras på detta sätt, trots att de inte servas ut direkt till klienter.
- Alla övriga filer som enbart är till för användning internt inom webbservern samlas i en underkatalog med namnet `WEB-INF` till webbapplikationens rotkatalog. Dit hör framför allt installationsbeskrivningen `web.xml`.
- Fristående objektkodsfiler för Java-klasser placeras relativt underkatalogen `classes` till katalogen `WEB-INF`, så att katalogen `classes` fungerar som rotkatalog för objektкод inom webbapplikationen. I vårt exempel fanns inga sådana klasser, så därför är katalogen tom.
- Biblioteket JSTL består av två JAR-filer: `jstl.jar` och `standard.jar`. Dessa placeras i underkatalogen `lib` till katalogen `WEB-INF`.
- JSP-kommandon deklarereras i en eller flera *TLD-filer* (*tag library descriptor*-filer). Dessa placeras på valfri plats under katalogen `WEB-INF` och måste pekas ut i installationsbeskrivningen. Vi har valt att samla dem i en underkatalog `tlds` under `WEB-INF`, vilket är en vanlig lösning. Knytningen mellan JSP-sidor och TLD-filer förklaras utförligare på nästa sida.

Deklaration av JSP-kommandon



Från bilden på föregående sida kan vi se att två JAR-filer installerats inom webbapplikationen games. Tillsammans innehåller dessa kommandobiblioteket **JSTL**, men det är inte tillräckligt att enbart installera dem i underkatalogen `lib` för att vi ska kunna använda de JSP-kommandon som ingår i JSTL.

Vi har redan nämnt att ett krav för att vi ska kunna använda ett JSP-kommando som tex `<c:forEach>` är att det lokala prefixet (här `c`) knutits till ett faktiskt kommandobibliotek. Det är vad vi gör med JSP-direktivet `taglib`:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

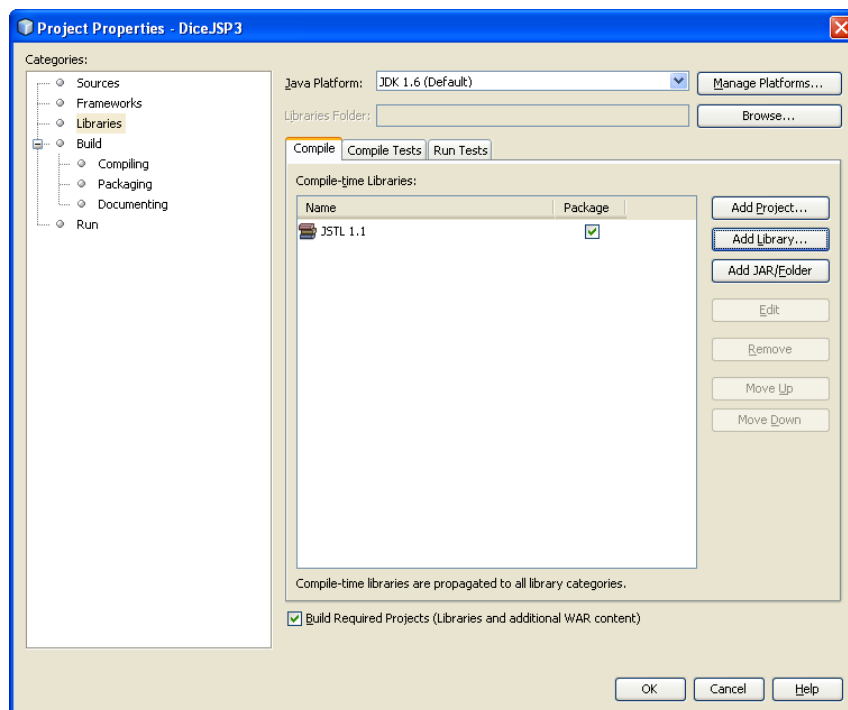
Med detta direktiv har vi knutit det lokala prefixet `c` till en specifik URI som fungerar som en adress som unikt identifierar biblioteket. Vi är emellertid inte färdiga ännu. Vi måste också ge webbservern deklarerationer av de JSP-kommandon som ingår i kommandobiblioteket, så att det går att avgöra om vi använder kommandona med rätt attribut, etc. Detta beskrivs i en *TLD-fil* (*tag library descriptor*) som distribueras med kommandobiblioteket. TLD-filen innehåller en deklARATION för varje kommando i biblioteket.

JSTL är ett stort bibliotek. De JSP-kommandon som ingår i biblioteket är indelade i underbibliotek som knyts till varsitt prefix. I vårt fall är deklARATIONerna för **Core**-biblioteket i JSTL samlade i filen `c.tld`. (Filens namn stämmer överens med att kommandon i biblioteket konventionsmässigt ges prefixet `c`.)

Installationsbeskrivningen `web.xml` innehåller ett element med namnet `jsp-config`. Inom detta element kan vi placera ett eller flera underelement med namnet `taglib`. Varje sådant element knyter samman en specifik URI som unikt identifierar ett bibliotek (samma URI som vi använder i direktivet `taglib`) med det fysiska filvägsnamnet för TLD-filen inom webbapplikationen. Här har vi valt att placera TLD-filen i en underkatalog `tlds` under katalogen `WEB-INF`. Det är inte nödvändigt att följa just den konventionen. Det viktiga är att `web.xml` pekar ut var TLD-filen faktiskt finns.

JSP-utveckling i NetBeans

Lägga till biblioteksfiler för JSTL



I verktyget **NetBeans** kan vi enkelt lägga till JSTL-biblioteket i ett projekt:

För att lägga till biblioteket JSTL, högerklicka på symbolen för projektet i vänsterspalten i verktyget och välj **Properties**.

Under kategorin *Libraries* visas vilka bibliotek och andra moduler som utnyttjas av projektet:

- Klicka på knappen **Add Library**.
- I den lista av bibliotek som nu visas, markera biblioteket **JSTL 1.1** och klicka på knappen **Add Library**. Biblioteket ska nu visas i listan i fönstret **Properties**.
- Klicka **OK** för att bekräfta de ändrade inställningarna.

De två JAR-filerna som ingår i JSTL ska nu synas under alternativet **Libraries** i vänsterspalten (dubbelklicka **Libraries** under projektets symbol om listan inte syns). Därmed kommer dessa biblioteksfiler automatiskt att tas med i WAR-filen för webbapplikationen.

Däremot måste vi själva placera TLD-filen `c.tld` på önskad plats under `WEB-INF` under alternativet **Web Pages** i vänsterspalten.

Kommandobibliotek i JSTL 1.2

Bibliotek	Lokalt prefix	Omfattar
Core	<code>c</code>	styrstrukturer för villkorlig och upprepad exekvering av webbinnehåll, felhantering, hantering av variabler, JavaBeans-komponenter och webbresurser
Functions	<code>fn</code>	vanliga nyttofunktioner, t ex stränghantering
I18N & Formatting	<code>fmt</code>	stöd för nationell anpassning i form av formatering och resurshantering
XML Processing	<code>x</code>	stöd för XML-bearbetning
Database Access	<code>sql</code>	stöd för databasåtkomst via JDBC

- JSTL är en obligatorisk del av JSP fr o m JSP 2.0
- JSTL 1.0 var ett frivilligt tillägg till JSP 1.2
- Kommandobiblioteket **Functions** ingick inte i JSTL 1.0.

JSTL består idag av fem kommandobibliotek:

- **Core** (kommandoprefix `c`) – stöd för bl a hantering av JSP-variabler och JavaBeans-komponenter, hantering av URL-adresserade webbresurser, styrstrukturer och felhantering
- **Functions** (kommandoprefix `fn`) – stöd för vanliga nyttofunktioner, t ex för stränghantering
- **I18N & Formatting** (kommandoprefix `fmt`) – stöd för nationell anpassning i form av formatering och resurshantering (den kryptiska förkortningen *I18N* står för *internationalization*)
- **XML Processing** (kommandoprefix `x`) – stöd för XML-bearbetning
- **Database Access** (kommandoprefix `sql`) – stöd för relationsdatabasåtkomst via JDBC

Den första versionen av JSTL, version 1.0, var ett frivilligt tillägg till JSP 1.2. I den ingick samtliga ovan nämnda kommandobibliotek utom **Functions**. Fr o m JSP 2.0 ingår JSTL som en obligatorisk del i JSP. Den version som ingick i JSP 2.0 var JSTL 1.1. Den nuvarande versionen, JSTL 1.2, är en obligatorisk del av JSP 2.1.